mg75gywbs

January 5, 2025

[ ]:
```python
simport numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

[ ]:
```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```python
data = '/content/drive/MyDrive/008 - My Projects/Loan Approval/Loan Sanction␣
↪CSV.csv'
df = pd.read_csv(data)
```

[ ]:
```python
'''Beginning by exploring the dataset'''
'''Understanding the structure of data, the Dtypes of variables available, and␣
↪the general patterns'''

df.head()
```

[ ]:
```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001015   Male     Yes          0      Graduate            No
1  LP001022   Male     Yes          1      Graduate            No
2  LP001031   Male     Yes          2      Graduate            No
3  LP001035   Male     Yes          2      Graduate            No
4  LP001051   Male      No          0  Not Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5720                  0       110.0             360.0
1             3076               1500       126.0             360.0
2             5000               1800       208.0             360.0
3             2340               2546       100.0             360.0
4             3276                  0        78.0             360.0

   Credit_History Property_Area
0             1.0         Urban
```

```
1             1.0         Urban
2             1.0         Urban
3             NaN         Urban
4             1.0         Urban
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             356 non-null    object
 2   Married            367 non-null    object
 3   Dependents         357 non-null    object
 4   Education          367 non-null    object
 5   Self_Employed      344 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

[ ]:
```
'''Point to be noted that feature '#Dependents' is in Object type even though␣
↪its values are '#Numeric', So need to convert that to int/float'''

'''Also i noticed certain '+' signs are visible along with some Numeric values␣
↪in '#Dependents' feature, so removing them FIRST'''

df['Dependents'] = df['Dependents'].str.replace('+','')
```

[ ]:
```
df['Dependents'].head(5)
```

[ ]:
```
0    0
1    1
2    2
3    2
4    0
Name: Dependents, dtype: object
```

[ ]:
```
'''Now converting from Object type to Numeric type(int/float)'''

df['Dependents'] = pd.to_numeric(df['Dependents'])
```

[ ]:
```
df['Dependents'].head(5)
```

[ ]:
```
0    0.0
1    1.0
2    2.0
3    2.0
4    0.0
Name: Dependents, dtype: float64
```

[ ]:
```
'''Checking for Dtype conversion and Null values'''

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             356 non-null    object
 2   Married            367 non-null    object
 3   Dependents         357 non-null    float64
 4   Education          367 non-null    object
 5   Self_Employed      344 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(4), int64(2), object(6)
memory usage: 34.5+ KB
```

[ ]:
```
'''Descriptive Statistics about our Dataset'''

df.describe()
```

[ ]:
```
       Dependents  ApplicantIncome  CoapplicantIncome  LoanAmount  \
count  357.000000       367.000000         367.000000  362.000000
mean     0.829132      4805.599455        1569.577657  136.132597
std      1.071302      4910.685399        2334.232099   61.366652
min      0.000000         0.000000           0.000000   28.000000
25%      0.000000      2864.000000           0.000000  100.250000
50%      0.000000      3786.000000        1025.000000  125.000000
75%      2.000000      5060.000000        2430.500000  158.000000
max      3.000000     72529.000000       24000.000000  550.000000
```

```
       Loan_Amount_Term  Credit_History
count        361.000000      338.000000
mean         342.537396        0.825444
std           65.156643        0.380150
min            6.000000        0.000000
25%          360.000000        1.000000
50%          360.000000        1.000000
75%          360.000000        1.000000
max          480.000000        1.000000
```

[ ]:
```
#As soon as we perform Exploratory and Descriptive analysis, we can now begin␣
↪Data Cleaning & Pre-processing.
```

[ ]:
```
'''Let's drop any duplicate entries and check the shape of our dataset'''

df.drop_duplicates()
df.shape
```

[ ]:
```
(367, 12)
```

[ ]:
```
'''Let's find Null/Missing values in our dataset(Column-wise)'''

df.isnull().sum()
```

[ ]:
```
Loan_ID               0
Gender               11
Married               0
Dependents           10
Education             0
Self_Employed        23
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            5
Loan_Amount_Term      6
Credit_History       29
Property_Area         0
dtype: int64
```

[ ]:
```
'''Total Number of Null values in our dataset'''

df.isnull().sum().sum()
```

[ ]:
```
84
```

[ ]:
```
'''We need to fill these missing values with the appropriate values, which␣
↪enables us analyse better insights from our dataset'''
```

```
'''The reason to fill Null/Missing values is that we can't analyse the data␣
↪without it'''
```

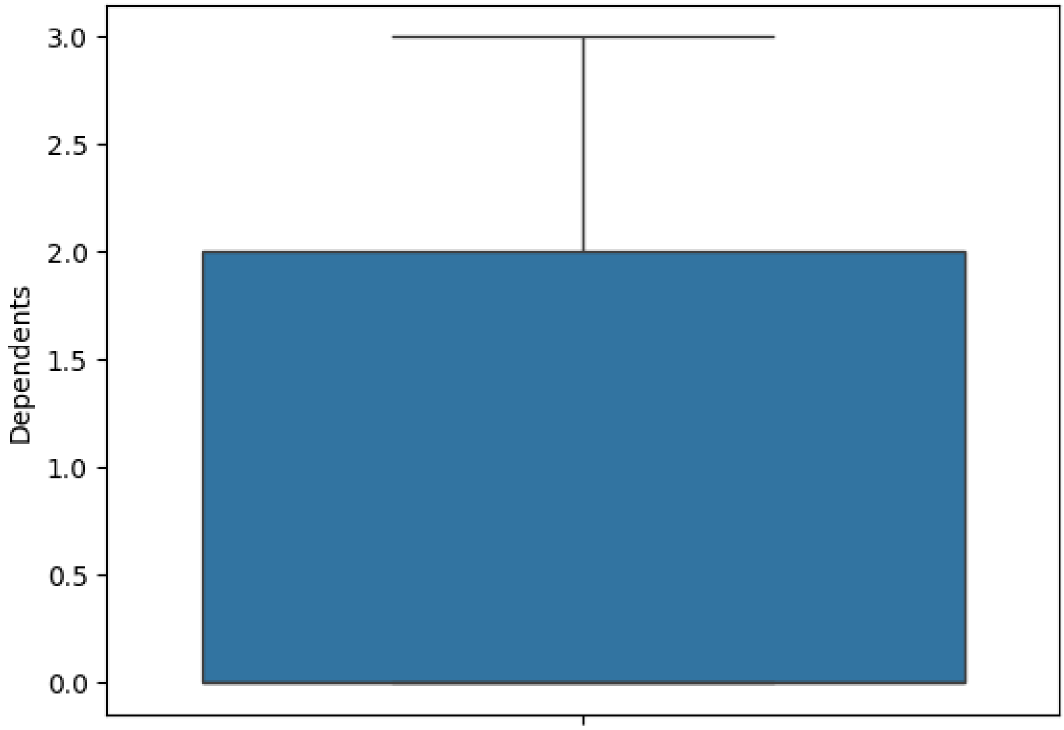
[ ]:
```
#For Categorical features, filling them with Mode.

#In 'Gender', Mode is 'Male'
#In 'Self_Employed', Mode is 'No'.

'''Feature - Gender'''
'''Feature - Self_Employed'''

df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])

df['Self_Employed'] = df['Self_Employed'].fillna(df['Self_Employed'].mode()[0])
```

[ ]:
```
#For Numerical features like

'''Dependents'''
'''LoanAmount'''
'''Loan_Amount_Term'''
'''Credit_History'''

#First calculate the percentage of Null values specifically column-wise.

df.isnull().sum()/df.shape[0]*100
```

[ ]:
```
Loan_ID              0.000000
Gender               0.000000
Married              0.000000
Dependents           2.724796
Education            0.000000
Self_Employed        0.000000
ApplicantIncome      0.000000
CoapplicantIncome    0.000000
LoanAmount           1.362398
Loan_Amount_Term     1.634877
Credit_History       7.901907
Property_Area        0.000000
dtype: float64
```

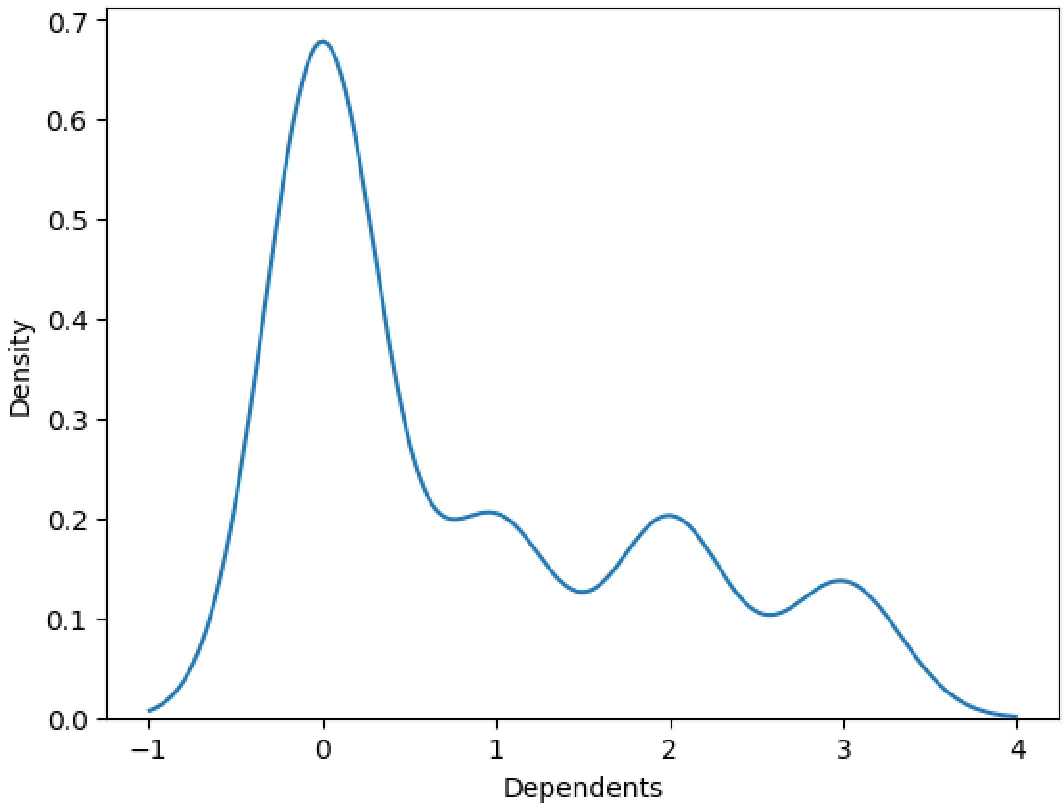
[ ]:
```
'''Feature - Dependents'''

# Drawing a boxplot of feature 'Dependents' for checking if there's any␣
↪outliers exist

sns.boxplot(df['Dependents'])
```

[ ]: <Axes: ylabel='Dependents'>



[ ]:
```
# Checking the distribution of feature 'Dependents'

sns.kdeplot(df['Dependents'])
```

[ ]: <Axes: xlabel='Dependents', ylabel='Density'>



[ ]:
```
# The feature 'Dependents' doesn't exist any outliers but as we can see, it is␣
↪positively skewed.

median_value = df['Dependents'].median()
median_value
```

[ ]:
```
0.0
```

[ ]:
```
# So filling the Null values in this feature by Median.

df['Dependents'] = df['Dependents'].fillna(median_value).astype(float)
```

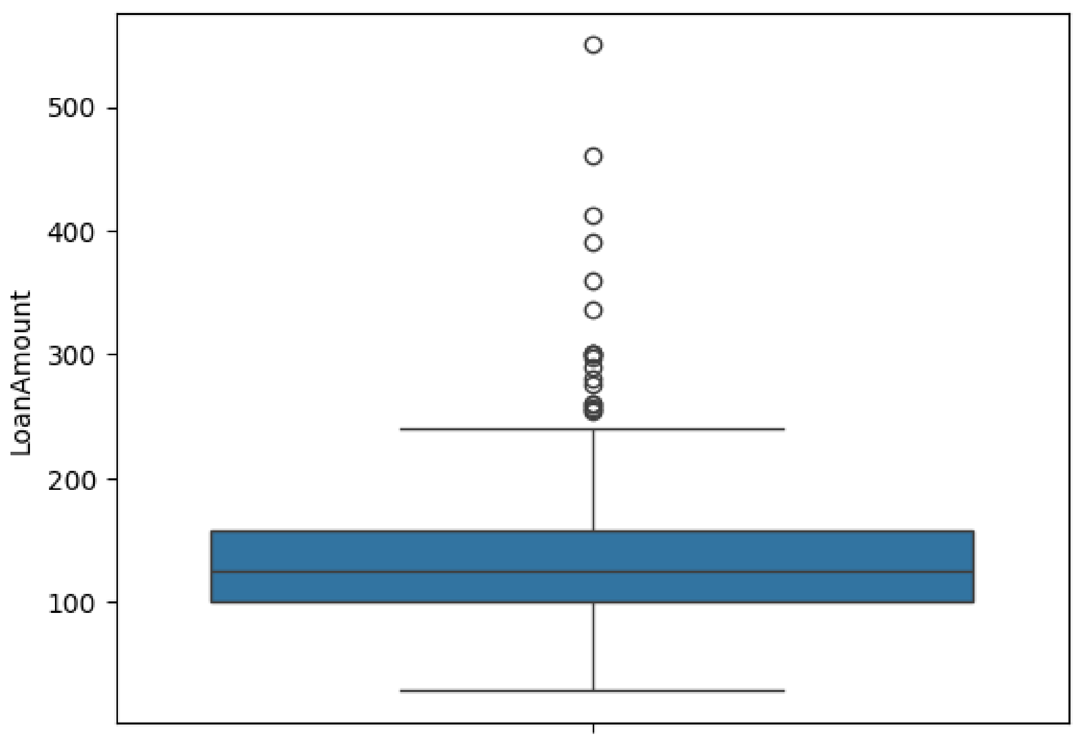
[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             367 non-null    object
```

```
 2   Married            367 non-null    object
 3   Dependents         367 non-null    float64
 4   Education          367 non-null    object
 5   Self_Employed      367 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(4), int64(2), object(6)
memory usage: 34.5+ KB
```

[ ]:
```
'''Feature - LoanAmount'''

# Drawing a boxplot of feature 'LoanAmount' for checking if there's any␣
↪outliers exist

sns.boxplot(df['LoanAmount'])
```

[ ]: <Axes: ylabel='LoanAmount'>



[ ]:
```
# Checking Median.

median_loan_amount = df['LoanAmount'].median()
median_loan_amount
```

[ ]: 125.0

[ ]:
```
# Filling Null values with Median first.

df['LoanAmount'] = df['LoanAmount'].fillna(median_loan_amount).astype(float)
```

[ ]:
```
'''Outliers handling in feature 'LoanAmount' by IQR Method.'''
```

[ ]:
```
# Calculating IQR for Feature 'LoanAmount'.

Q1 = df['LoanAmount'].quantile(0.25)
print(f"Q1 is {Q1}")

Q3 = df['LoanAmount'].quantile(0.75)
print(f"Q3 is {Q3}")
```

```
Q1 is 101.0
Q3 is 157.5
```

[ ]:
```
IQR = Q3 - Q1
print(f"IQR is {IQR}")
```

```
IQR is 56.5
```

[ ]:
```
# Defining the outlier boundaries.

lower_bound = Q1 - 1.5 * IQR
print(lower_bound)

upper_bound = Q3 + 1.5 * IQR
print(upper_bound)
```

```
16.25
242.25
```

[ ]:
```
# Identifying outliers.

# Our dataset lies between 16.25 and 242.25 as per IQR Method, therefore any␣
↪value below 16.25 and beyond 242.25 is considered as an outlier in feature␣
↪'LoanAmount'.
```

```
outliers = df[(df['LoanAmount'] < lower_bound) | (df['LoanAmount'] >
 ↪upper_bound)]
outliers
```

[ ]:

```
       Loan_ID  Gender Married  Dependents Education Self_Employed  \
8     LP001059    Male     Yes         2.0  Graduate            No
18    LP001108    Male     Yes         0.0  Graduate            No
24    LP001149    Male     Yes         0.0  Graduate            No
27    LP001169    Male     Yes         0.0  Graduate            No
81    LP001428    Male     Yes         3.0  Graduate            No
83    LP001446    Male     Yes         0.0  Graduate            No
91    LP001483    Male     Yes         3.0  Graduate            No
96    LP001500    Male     Yes         1.0  Graduate            No
124   LP001655  Female      No         0.0  Graduate            No
143   LP001791    Male     Yes         0.0  Graduate           Yes
144   LP001794    Male     Yes         2.0  Graduate           Yes
189   LP002059    Male     Yes         2.0  Graduate            No
194   LP002077    Male     Yes         1.0  Graduate            No
284   LP002570  Female     Yes         2.0  Graduate            No
285   LP002572    Male     Yes         1.0  Graduate            No
331   LP002825    Male     Yes         3.0  Graduate            No
345   LP002878    Male     Yes         3.0  Graduate            No
350   LP002899    Male     Yes         2.0  Graduate            No

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
8              13633                  0       280.0             240.0
18              9226               7916       300.0             360.0
24              5400               4380       290.0             360.0
27              7500               3750       275.0             360.0
81             72529                  0       360.0             360.0
83              8449                  0       257.0             360.0
91             13518                  0       390.0             360.0
96              3333               4200       256.0             360.0
124            12500                  0       300.0             360.0
143            32000                  0       550.0             360.0
144            10890                  0       260.0              12.0
189             7874               3967       336.0             360.0
194            10000               2690       412.0             360.0
284            10000              11666       460.0             360.0
285             8750                  0       297.0             360.0
331             9699                  0       300.0             360.0
345             8334                  0       260.0             360.0
350             8667                  0       254.0             360.0

     Credit_History Property_Area
8               1.0         Urban
18              1.0         Urban
```

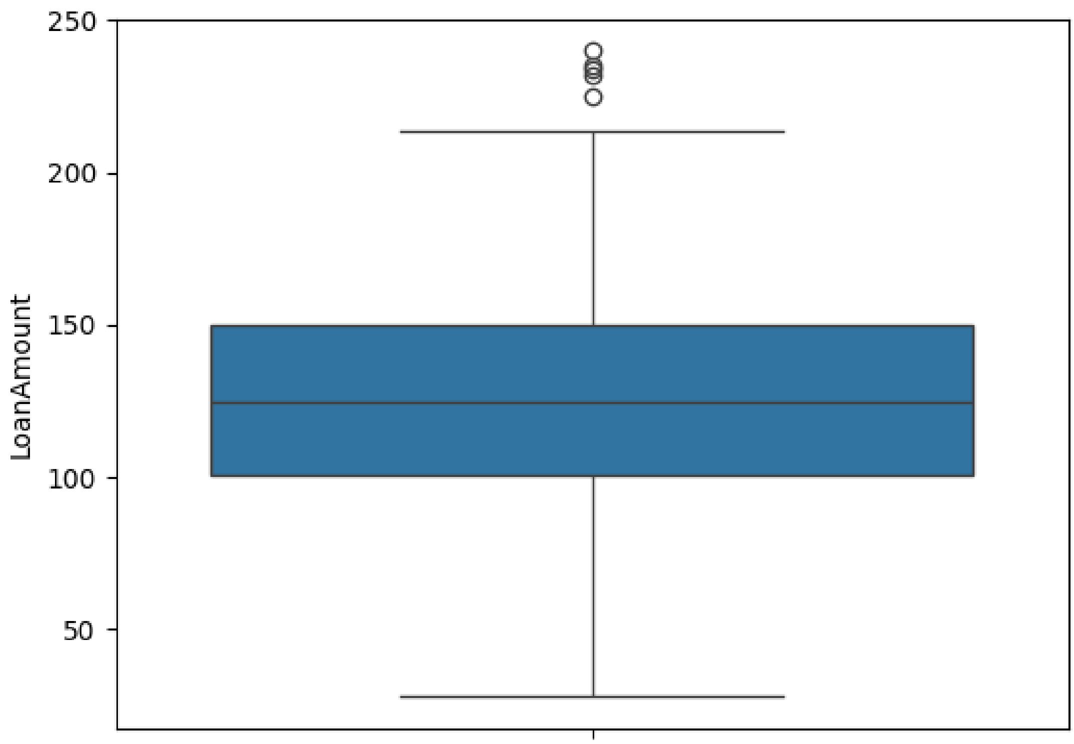
```
24              1.0        Urban
27              1.0        Urban
81              1.0        Urban
83              1.0        Rural
91              1.0        Rural
96              1.0        Urban
124             0.0        Urban
143             NaN    Semiurban
144             1.0        Rural
189             1.0        Rural
194             1.0    Semiurban
284             1.0        Urban
285             1.0        Urban
331             1.0        Urban
345             1.0        Urban
350             1.0        Rural
```

[ ]:
```
# Replacing Outliers with Median.

df['LoanAmount'] = np.where((df['LoanAmount'] < lower_bound) |
 ↪(df['LoanAmount'] > upper_bound), median_loan_amount, df['LoanAmount'])
print(df['LoanAmount'])
```

```
0      110.0
1      126.0
2      208.0
3      100.0
4       78.0
       …
362    113.0
363    115.0
364    126.0
365    158.0
366     98.0
Name: LoanAmount, Length: 367, dtype: float64
```

[ ]:
```
# We see that there are still a few outliers exists.

sns.boxplot(df['LoanAmount'])
```

[ ]: <Axes: ylabel='LoanAmount'>




```
[ ]: # Calculating IQR for 'LoanAmount' again after the previous transformations.

     Q1 = df['LoanAmount'].quantile(0.25)
     print(f"Q1 is {Q1}")

     Q3 = df['LoanAmount'].quantile(0.75)
     print(f"Q3 is {Q3}")
```

```
Q1 is 101.0
Q3 is 150.0
```

```
[ ]: IQR = Q3 - Q1
     print(f"IQR is {IQR}")
```

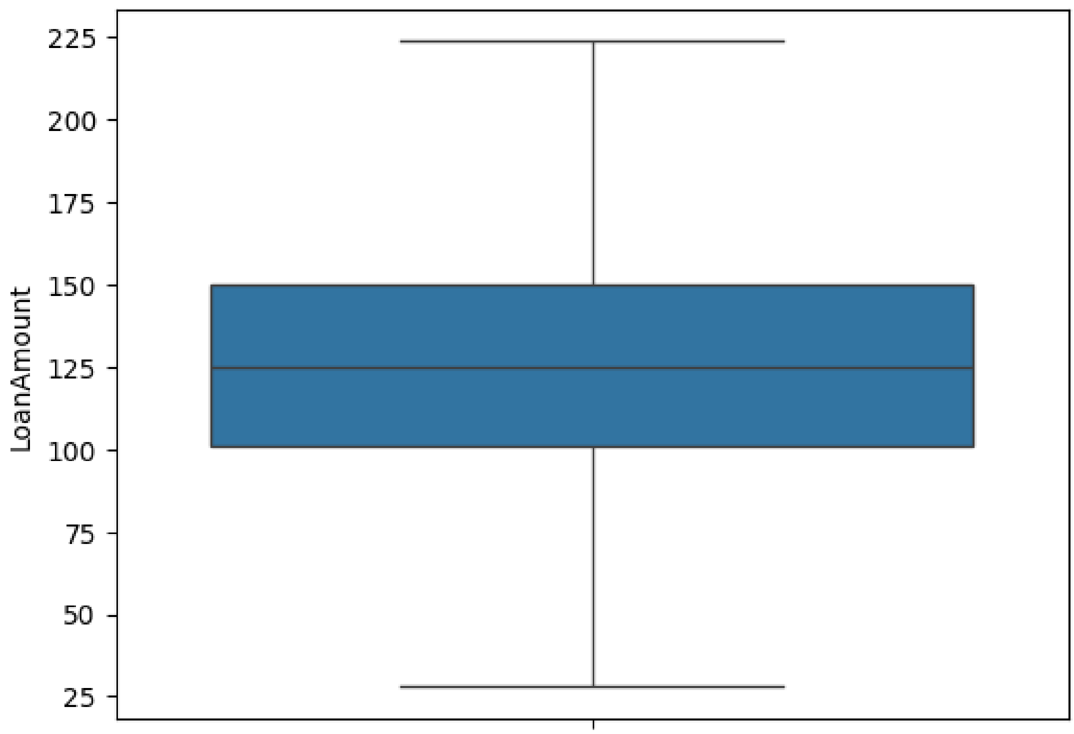
```
IQR is 49.0
```

```
[ ]: # Defining new outlier boundaries.

     lower_bound = Q1 - 1.5 * IQR
     print(lower_bound)

     upper_bound = Q3 + 1.5 * IQR
     print(upper_bound)
```

```
27.5
223.5
```

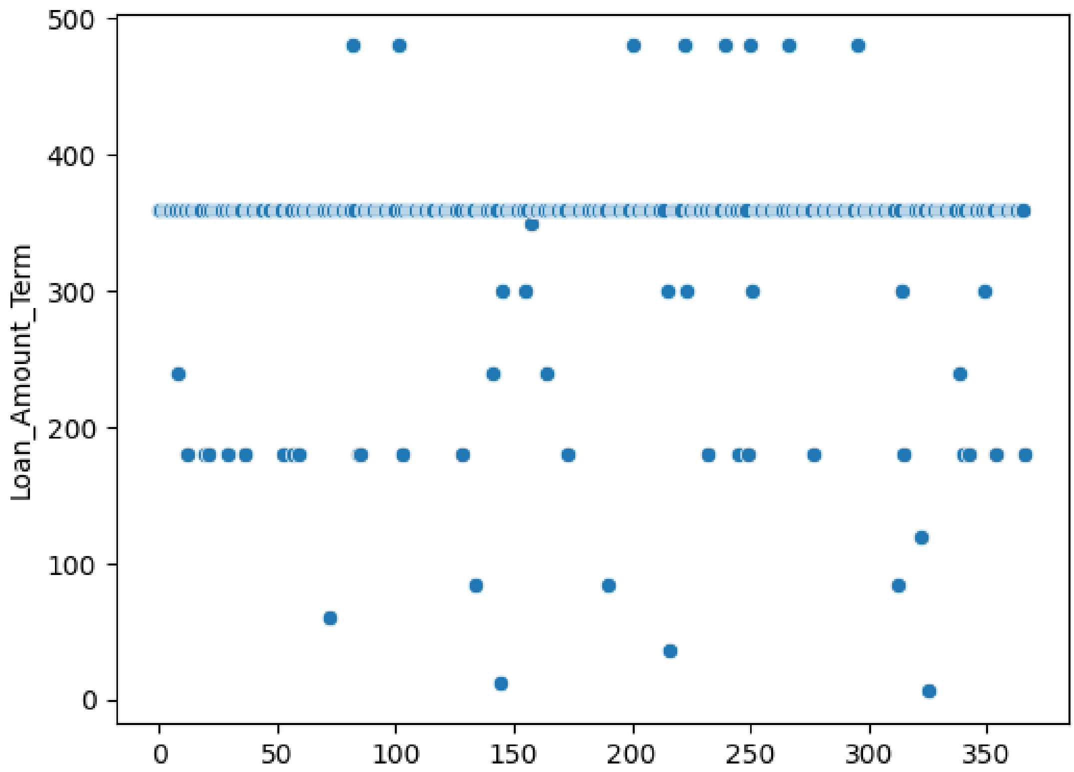
[ ]:
```python
# Now, instead of replacing with Median, Capping the outliers to boundary␣
 ↪values.

df['LoanAmount'] = np.where(df['LoanAmount'] < lower_bound, lower_bound,␣
 ↪df['LoanAmount'])
df['LoanAmount'] = np.where(df['LoanAmount'] > upper_bound, upper_bound,␣
 ↪df['LoanAmount'])
```

[ ]:
```python
# Checking our Distribution again.

sns.boxplot(df['LoanAmount'])
plt.show()
```



[ ]:
```python
'''Feature - Loan_Amount_Term'''

# Drawing a scatterplot of feature 'Loan_Amount_Term' for checking if there's␣
 ↪any outliers exist

sns.scatterplot(df['Loan_Amount_Term'])
```

[ ]: <Axes: ylabel='Loan_Amount_Term'>



[ ]:
```
'''Outliers handling in feature 'Loan_Amount_Term' by IQR Method.'''
```

[ ]:
```
# Calculating IQR for 'Loan_Amount_Term'.

Q1 = df['Loan_Amount_Term'].quantile(0.25)
print(f"Q1 is {Q1}")

Q3 = df['Loan_Amount_Term'].quantile(0.75)
print(f"Q3 is {Q3}")
```

```
Q1 is 360.0
Q3 is 360.0
```

[ ]:
```
IQR = Q3 - Q1
print(f"IQR is {IQR}")
```

```
IQR is 0.0
```

[ ]:
```
# Defining the outlier boundaries.
```

```
lower_bound = Q1 - 1.5 * IQR
print(lower_bound)

upper_bound = Q3 + 1.5 * IQR
print(upper_bound)
```

```
360.0
360.0
```

[ ]:
```
# Identifying outliers.

# Our dataset lies at 360 as per IQR Method, therefore any value below and␣
↪beyond 360.0 is considered as an outlier in feature 'Loan_Amount_Term'.

outliers = df[(df['Loan_Amount_Term'] < lower_bound) | (df['Loan_Amount_Term']␣
↪> upper_bound)]
outliers
```

[ ]:
```
      Loan_ID  Gender Married  Dependents     Education Self_Employed  \
8    LP001059    Male     Yes         2.0      Graduate            No
12   LP001083    Male      No         3.0      Graduate            No
19   LP001115    Male      No         0.0      Graduate            No
21   LP001124  Female      No         3.0  Not Graduate            No
29   LP001176    Male      No         0.0      Graduate            No
36   LP001208    Male     Yes         2.0      Graduate            No
53   LP001298    Male     Yes         2.0      Graduate            No
57   LP001321    Male     Yes         2.0      Graduate            No
59   LP001324    Male     Yes         3.0      Graduate            No
72   LP001375    Male     Yes         1.0      Graduate            No
82   LP001445    Male     Yes         2.0  Not Graduate            No
84   LP001450    Male     Yes         0.0      Graduate            No
85   LP001452    Male     Yes         2.0      Graduate            No
101  LP001542  Female     Yes         0.0      Graduate            No
103  LP001548    Male     Yes         2.0  Not Graduate            No
128  LP001667    Male      No         0.0      Graduate            No
134  LP001737    Male      No         0.0      Graduate            No
141  LP001787    Male     Yes         3.0      Graduate            No
144  LP001794    Male     Yes         2.0      Graduate           Yes
145  LP001797  Female      No         0.0      Graduate            No
155  LP001857    Male      No         0.0  Not Graduate           Yes
157  LP001867    Male     Yes         0.0      Graduate            No
164  LP001921    Male      No         1.0      Graduate            No
173  LP001979    Male      No         0.0      Graduate            No
190  LP002062  Female     Yes         1.0      Graduate            No
200  LP002105    Male     Yes         0.0      Graduate           Yes
215  LP002184    Male     Yes         0.0  Not Graduate            No
216  LP002186    Male     Yes         0.0  Not Graduate            No
```

222	LP002245	Male	Yes	2.0	Not Graduate	No
223	LP002253	Female	No	1.0	Graduate	No
232	LP002306	Male	Yes	0.0	Graduate	No
239	LP002329	Male	No	0.0	Graduate	No
245	LP002355	Male	Yes	0.0	Graduate	No
249	LP002376	Male	No	0.0	Graduate	No
250	LP002383	Male	Yes	3.0	Graduate	No
251	LP002385	Male	Yes	0.0	Graduate	No
266	LP002442	Female	Yes	1.0	Not Graduate	No
277	LP002550	Female	No	0.0	Graduate	No
295	LP002612	Female	Yes	0.0	Graduate	No
312	LP002754	Male	No	0.0	Graduate	No
314	LP002760	Female	No	0.0	Graduate	No
315	LP002766	Female	Yes	0.0	Graduate	No
322	LP002790	Male	Yes	3.0	Graduate	No
325	LP002802	Male	No	0.0	Graduate	No
338	LP002857	Male	Yes	1.0	Graduate	Yes
340	LP002860	Male	Yes	0.0	Graduate	Yes
342	LP002869	Male	Yes	3.0	Not Graduate	No
349	LP002891	Male	Yes	0.0	Graduate	Yes
354	LP002921	Male	Yes	3.0	Not Graduate	No
366	LP002989	Male	No	0.0	Graduate	Yes

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term \
8	13633	0	125.0	240.0
12	4166	0	40.0	180.0
19	1300	3470	100.0	180.0
21	2083	0	28.0	180.0
29	2942	2382	125.0	180.0
36	7350	4029	185.0	180.0
53	4116	1000	30.0	180.0
57	3613	3539	134.0	180.0
59	4720	0	90.0	180.0
72	4083	1775	139.0	60.0
82	4136	0	149.0	480.0
84	4456	0	131.0	180.0
85	4635	8000	102.0	180.0
101	2262	0	125.0	480.0
103	2687	0	50.0	180.0
128	3073	0	70.0	180.0
134	4000	0	83.0	84.0
141	3089	2999	100.0	240.0
144	10890	0	125.0	12.0
145	12941	0	150.0	300.0
155	1599	2474	125.0	300.0
157	4333	2291	133.0	350.0
164	3180	2370	80.0	240.0

173	3017	2845	159.0	180.0
190	4333	0	132.0	84.0
200	8706	0	108.0	480.0
215	2914	2130	150.0	300.0
216	2747	2458	118.0	36.0
222	2896	0	80.0	480.0
223	5062	0	152.0	300.0
232	1173	1594	28.0	180.0
239	4333	0	66.0	480.0
245	3186	3145	150.0	180.0
249	2925	0	40.0	180.0
250	3242	437	142.0	480.0
251	3863	0	70.0	300.0
266	3835	1400	112.0	480.0
277	5769	0	110.0	180.0
295	2666	0	84.0	480.0
312	2066	2108	104.0	84.0
314	3767	0	134.0	300.0
315	7859	879	165.0	180.0
322	3400	0	80.0	120.0
325	2875	2416	95.0	6.0
338	2360	3355	87.0	240.0
340	2623	4831	122.0	180.0
342	3522	0	81.0	180.0
349	2500	296	137.0	300.0
354	5316	187	158.0	180.0
366	9200	0	98.0	180.0

	Credit_History	Property_Area
8	1.0	Urban
12	NaN	Urban
19	1.0	Semiurban
21	1.0	Urban
29	1.0	Urban
36	1.0	Urban
53	1.0	Urban
57	1.0	Semiurban
59	1.0	Semiurban
72	1.0	Urban
82	0.0	Rural
84	0.0	Semiurban
85	1.0	Rural
101	0.0	Semiurban
103	1.0	Rural
128	1.0	Urban
134	1.0	Urban
141	1.0	Rural

```
144             1.0        Rural
145             1.0        Urban
155             1.0    Semiurban
157             1.0        Rural
164             NaN        Rural
173             0.0        Urban
190             1.0        Rural
200             1.0        Rural
215             1.0        Urban
216             1.0    Semiurban
222             1.0        Urban
223             1.0        Rural
232             1.0        Rural
239             1.0        Urban
245             0.0    Semiurban
249             1.0        Rural
250             0.0        Urban
251             1.0    Semiurban
266             0.0        Urban
277             1.0    Semiurban
295             1.0    Semiurban
312             1.0        Urban
314             1.0        Urban
315             1.0    Semiurban
322             1.0        Urban
325             0.0    Semiurban
338             1.0        Rural
340             1.0    Semiurban
342             1.0        Rural
349             1.0        Rural
354             0.0    Semiurban
366             1.0        Rural
```

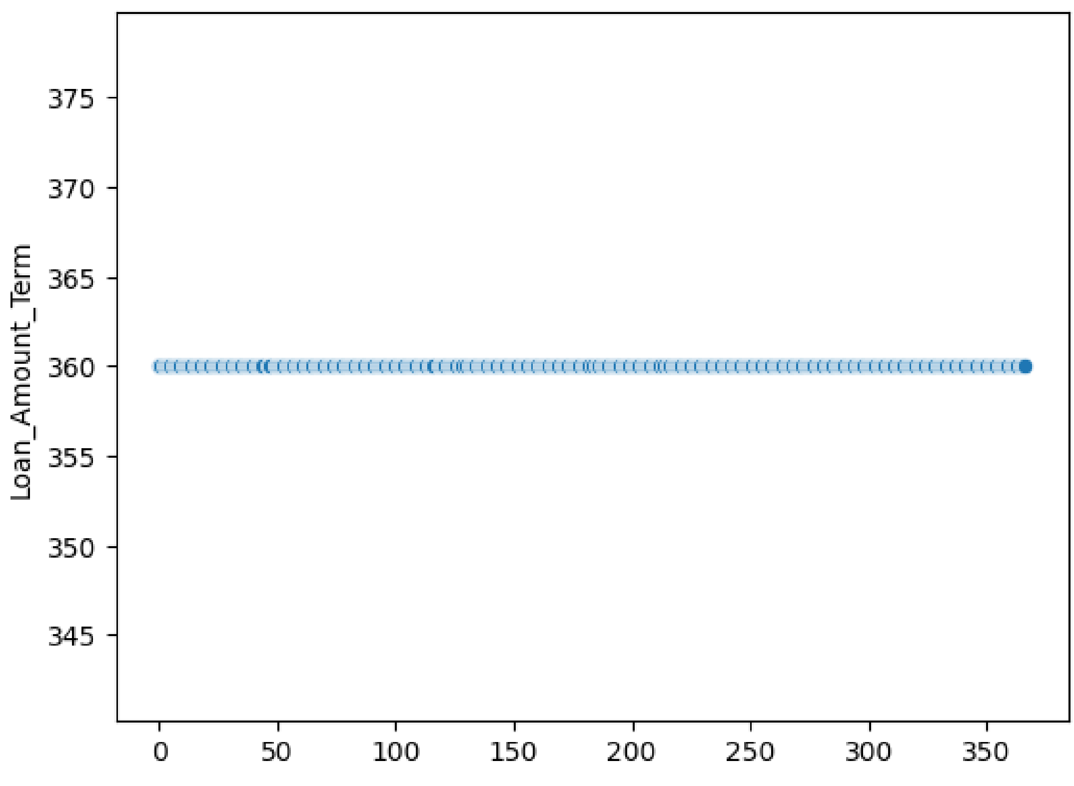
[ ]:
```
# Checking Median.

median_loan_term = df['Loan_Amount_Term'].median()
median_loan_term
```

[ ]: 360.0

[ ]:
```
# Replacing outliers with Median.

df['Loan_Amount_Term'] = np.where((df['Loan_Amount_Term'] < lower_bound) |␣
 ↪(df['Loan_Amount_Term'] > upper_bound), median_loan_term,␣
 ↪df['Loan_Amount_Term'])
print(df['Loan_Amount_Term'])
```

```
0      360.0
1      360.0
2      360.0
3      360.0
4      360.0
       …
362    360.0
363    360.0
364    360.0
365    360.0
366    360.0
Name: Loan_Amount_Term, Length: 367, dtype: float64
```

[ ]:
```python
# Visualizing the distribution after handling outliers.

sns.scatterplot(df['Loan_Amount_Term'])
plt.show()
```



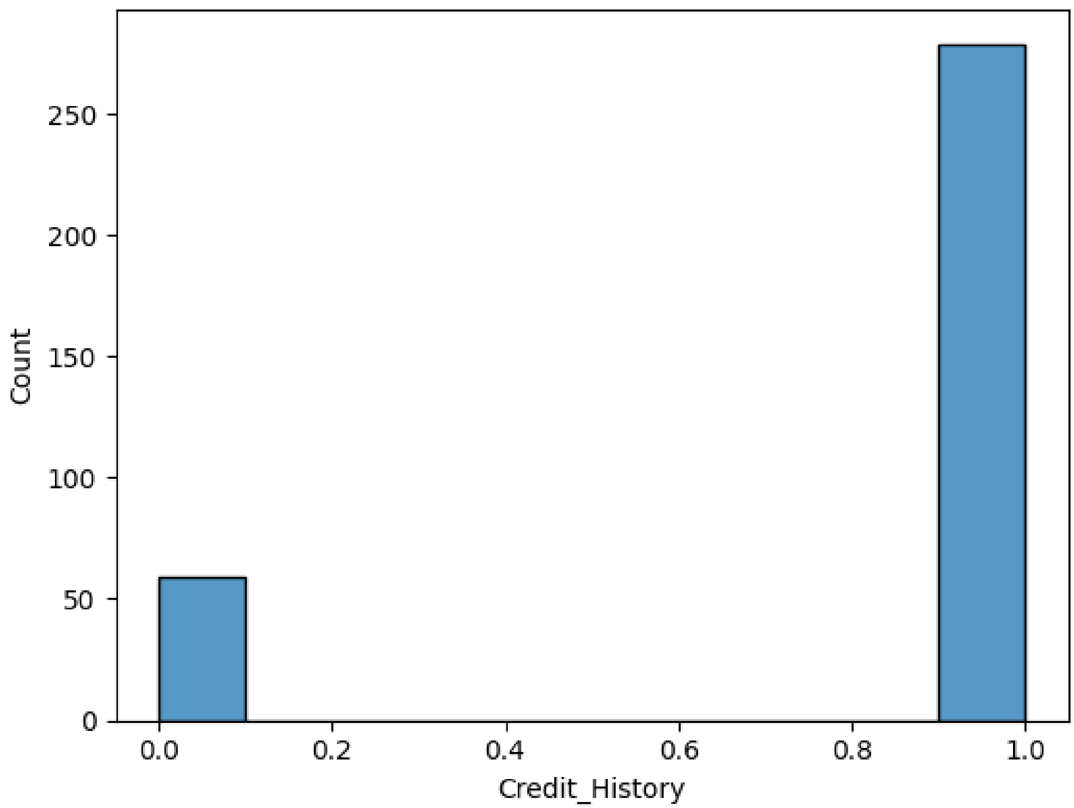
[ ]:
```python
# As the Outliers have been handled, we need to fill missing values in feature
↪'Loan_Amount_Term' by Median.
```

```
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(median_loan_term).
↪astype(float)
```

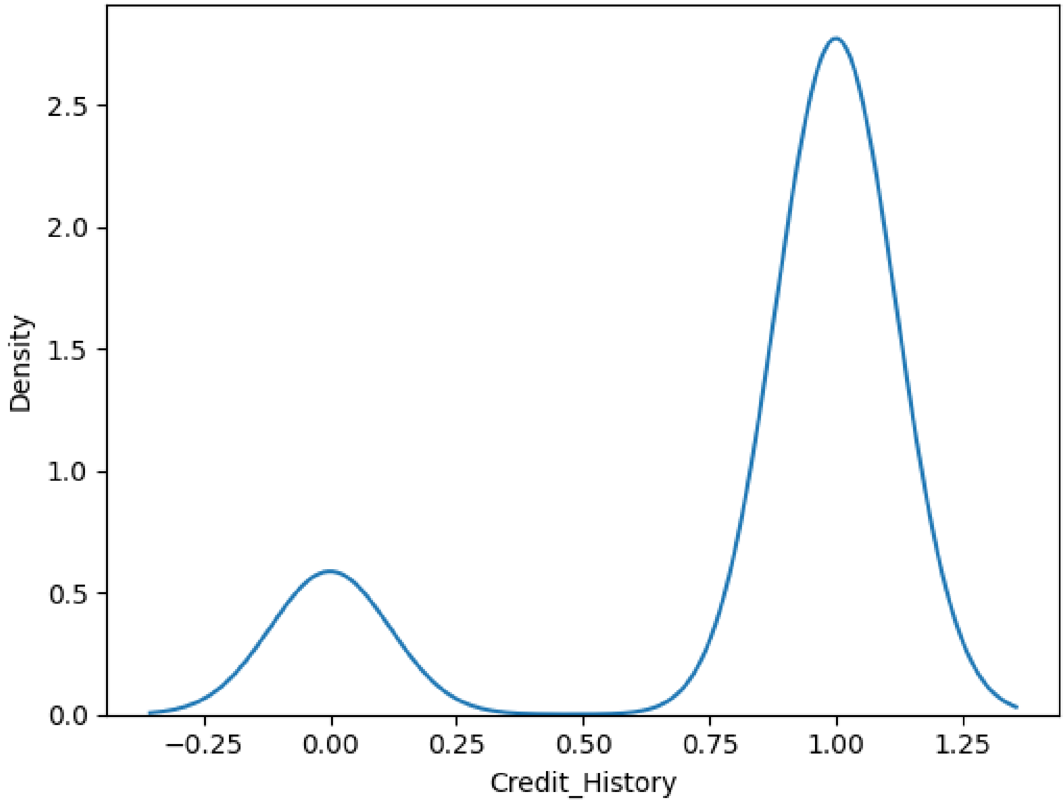
[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             367 non-null    object
 2   Married            367 non-null    object
 3   Dependents         367 non-null    float64
 4   Education          367 non-null    object
 5   Self_Employed      367 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         367 non-null    float64
 9   Loan_Amount_Term   367 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(4), int64(2), object(6)
memory usage: 34.5+ KB
```

[ ]:
```
'''Feature - Credit_History'''

# Drawing a histplot of feature 'Credit_History' for checking if there's any␣
↪outliers exist

sns.histplot(df['Credit_History'])
plt.show()
```



[ ]:
```
# Checking the distribution of feature 'Credit_History'

sns.kdeplot(df['Credit_History'])
plt.show()
```



[ ]:
```
df['Credit_History'].value_counts()
```

[ ]:
```
Credit_History
1.0    279
0.0     59
Name: count, dtype: int64
```

[ ]:
```
# The feature "Credit_History" doesn't have any outliers but as we can see, it␣
↪is negatively skewed.

credit_median = df['Credit_History'].median()
credit_median
```

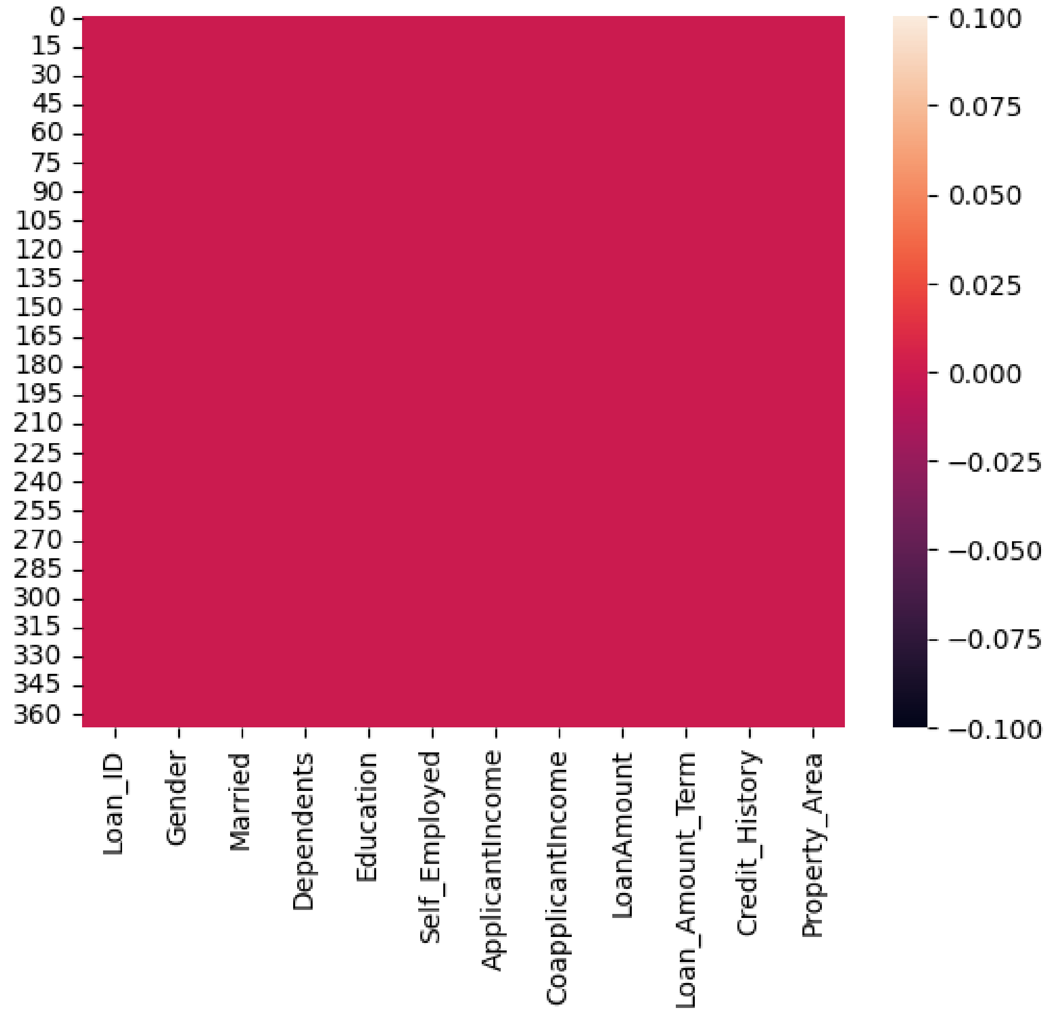
[ ]:
```
1.0
```

[ ]:
```
# So filling the Null values in this feature by Median.

df['Credit_History'] = df['Credit_History'].fillna(credit_median).astype(int)
```

[ ]:
```
df.info()
```

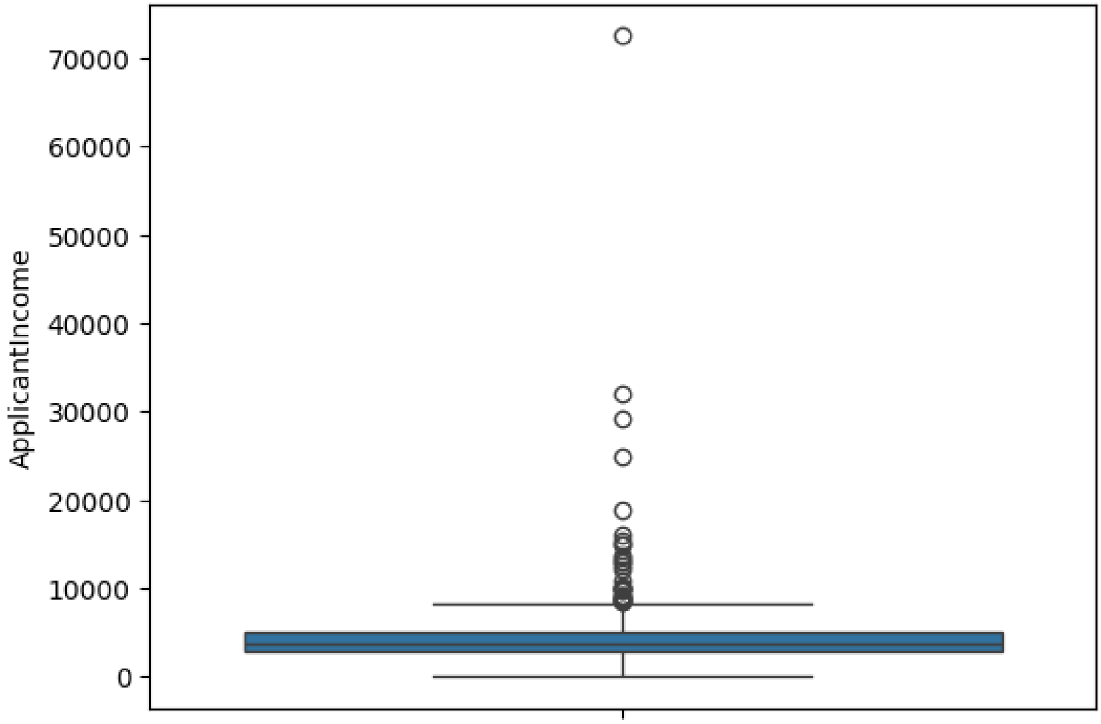
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             367 non-null    object
 2   Married            367 non-null    object
 3   Dependents         367 non-null    float64
 4   Education          367 non-null    object
 5   Self_Employed      367 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         367 non-null    float64
 9   Loan_Amount_Term   367 non-null    float64
 10  Credit_History     367 non-null    int64
 11  Property_Area      367 non-null    object
dtypes: float64(3), int64(3), object(6)
memory usage: 34.5+ KB
```

[ ]:
```
'''Let's draw a HEATMAP to ensure all Null values has been handled'''
sns.heatmap(df.isnull())
plt.show()
```



[ ]:
```
# NOTE

# As all the Outliers and Null values in above features has been handled, We
  ↪still have two Numerical features left to check at least for Outliers.
# The features are 'ApplicantIncome' & 'CoapplicantIncome'.
```

[ ]:
```
'''Feature - ApplicantIncome'''

# Drawing a boxplot of feature 'ApplicantIncome' for checking if there's any
  ↪outliers exist

sns.boxplot(df['ApplicantIncome'])
plt.show()
```




```
[ ]: '''Outliers handling in feature 'ApplicantIncome' by IQR Method.'''
```

```
[ ]: # Calculating IQR for 'ApplicantIncome'.

Q1 = df['ApplicantIncome'].quantile(0.25)
print(f"Q1 is {Q1}")

Q3 = df['ApplicantIncome'].quantile(0.75)
print(f"Q3 is {Q3}")
```

```
Q1 is 2864.0
Q3 is 5060.0
```

```
[ ]: IQR = Q3 - Q1
print(f"IQR is {IQR}")
```

```
IQR is 2196.0
```

```
[ ]: # Defining the outlier boundaries.

lower_bound = Q1 - 1.5 * IQR
print(lower_bound)
```

```python
upper_bound = Q3 + 1.5 * IQR
print(upper_bound)
```

```
-430.0
8354.0
```

[ ]:
```python
# Identifying outliers.

# Our dataset lies between -430 and 8354 as per IQR Method, therefore any value␣
↪below -430 and beyond 8354 is considered as an outlier in feature␣
↪'LoanAmount'.

outliers = df[(df['ApplicantIncome'] < lower_bound) | (df['ApplicantIncome'] >␣
↪upper_bound)]
outliers
```

[ ]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	\
8	LP001059	Male	Yes	2.0	Graduate	No	
13	LP001094	Male	Yes	2.0	Graduate	No	
18	LP001108	Male	Yes	0.0	Graduate	No	
81	LP001428	Male	Yes	3.0	Graduate	No	
83	LP001446	Male	Yes	0.0	Graduate	No	
91	LP001483	Male	Yes	3.0	Graduate	No	
98	LP001517	Male	Yes	3.0	Graduate	No	
124	LP001655	Female	No	0.0	Graduate	No	
143	LP001791	Male	Yes	0.0	Graduate	Yes	
144	LP001794	Male	Yes	2.0	Graduate	Yes	
145	LP001797	Female	No	0.0	Graduate	No	
147	LP001817	Male	No	0.0	Not Graduate	Yes	
179	LP002017	Male	Yes	3.0	Graduate	No	
184	LP002045	Male	Yes	3.0	Graduate	No	
187	LP002056	Male	Yes	2.0	Graduate	No	
188	LP002057	Male	Yes	0.0	Not Graduate	No	
194	LP002077	Male	Yes	1.0	Graduate	No	
200	LP002105	Male	Yes	0.0	Graduate	Yes	
230	LP002294	Male	No	0.0	Graduate	No	
247	LP002360	Male	Yes	0.0	Graduate	No	
263	LP002433	Male	Yes	1.0	Graduate	No	
272	LP002485	Male	No	1.0	Graduate	No	
279	LP002553	Male	No	0.0	Graduate	No	
283	LP002568	Male	No	0.0	Not Graduate	No	
284	LP002570	Female	Yes	2.0	Graduate	No	
285	LP002572	Male	Yes	1.0	Graduate	No	
302	LP002654	Female	No	0.0	Graduate	Yes	
323	LP002791	Male	No	1.0	Graduate	No	
331	LP002825	Male	Yes	3.0	Graduate	No	
350	LP002899	Male	Yes	2.0	Graduate	No	

```
360  LP002965  Female     Yes          0.0      Graduate            No
366  LP002989    Male      No          0.0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
8              13633                  0       125.0             360.0
13             12173                  0       166.0             360.0
18              9226               7916       125.0             360.0
81             72529                  0       125.0             360.0
83              8449                  0       125.0             360.0
91             13518                  0       125.0             360.0
98              9719                  0        61.0             360.0
124            12500                  0       125.0             360.0
143            32000                  0       125.0             360.0
144            10890                  0       125.0             360.0
145            12941                  0       150.0             360.0
147             8703                  0       199.0             360.0
179            15312                  0       187.0             360.0
184            10166                750       150.0             360.0
187             9167                  0       223.5             360.0
188            13083                  0       125.0             360.0
194            10000               2690       125.0             360.0
200             8706                  0       108.0             360.0
230            14911              14507       130.0             360.0
247            10000                  0       125.0             360.0
263            18840                  0       223.5             360.0
272            24797                  0       223.5             360.0
279            29167                  0       185.0             360.0
283             9000                  0       122.0             360.0
284            10000              11666       125.0             360.0
285             8750                  0       125.0             360.0
302            14987                  0       177.0             360.0
323            16000               5000        40.0             360.0
331             9699                  0       125.0             360.0
350             8667                  0       125.0             360.0
360             8550               4255        96.0             360.0
366             9200                  0        98.0             360.0

     Credit_History Property_Area
8                 1         Urban
13                0     Semiurban
18                1         Urban
81                1         Urban
83                1         Rural
91                1         Rural
98                1         Urban
124               0         Urban
143               1     Semiurban
```

```
144                 1         Rural
145                 1         Urban
147                 0         Rural
179                 1         Urban
184                 1         Urban
187                 1     Semiurban
188                 1         Rural
194                 1     Semiurban
200                 1         Rural
230                 1     Semiurban
247                 1         Urban
263                 1         Rural
272                 1     Semiurban
279                 1     Semiurban
283                 1         Rural
284                 1         Urban
285                 1         Urban
302                 1         Rural
323                 1     Semiurban
331                 1         Urban
350                 1         Rural
360                 1         Urban
366                 1         Rural
```

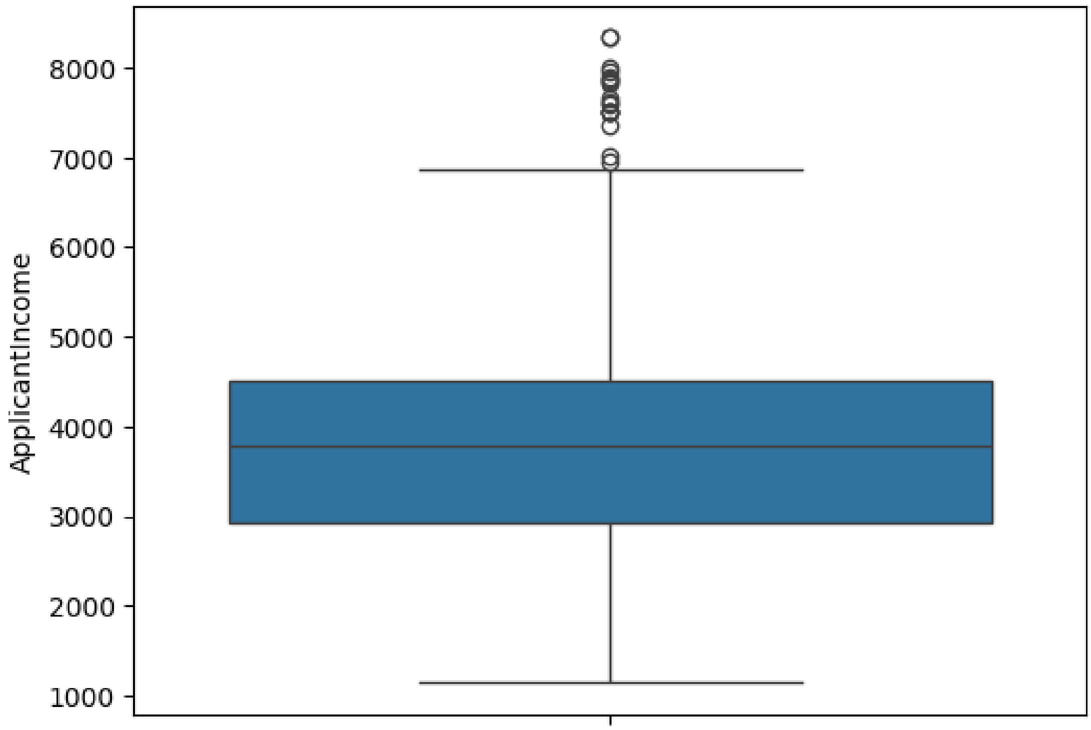
[ ]:
```
# Checking Median.

median_applicant_income = df['ApplicantIncome'].median()
median_applicant_income
```

[ ]: 3786.0

[ ]:
```
# Replacing Outliers in 'ApplicantIncome' with Median
# and
# Replacing 'ApplicantIncome' < 1000 with Median too.
# (Reason being two values have 0 income and remaining two values have barely␣
 ↪any income, which is practically not a possible income for a primary loan␣
 ↪applicant)

df['ApplicantIncome'] = np.where((df['ApplicantIncome'] < lower_bound) |␣
 ↪(df['ApplicantIncome'] > upper_bound), median_applicant_income,␣
 ↪df['ApplicantIncome'])

df['ApplicantIncome'] = np.where(df['ApplicantIncome'] < 1000 ,␣
 ↪median_applicant_income, df['ApplicantIncome'])
```

[ ]:
```
# We see that there are still a few outliers exists.
```

```
sns.boxplot(df['ApplicantIncome'])
plt.show()
```




```
[ ]: # Calculate the IQR for 'ApplicantIncome' again after the previous
     ↪transformations.

     Q1 = df['ApplicantIncome'].quantile(0.25)
     print(f"Q1 is {Q1}")

     Q3 = df['ApplicantIncome'].quantile(0.75)
     print(f"Q3 is {Q3}")
```

```
Q1 is 2915.5
Q3 is 4506.5
```

```
[ ]: IQR = Q3 - Q1
     print(f"IQR is {IQR}")
```

```
IQR is 1591.0
```

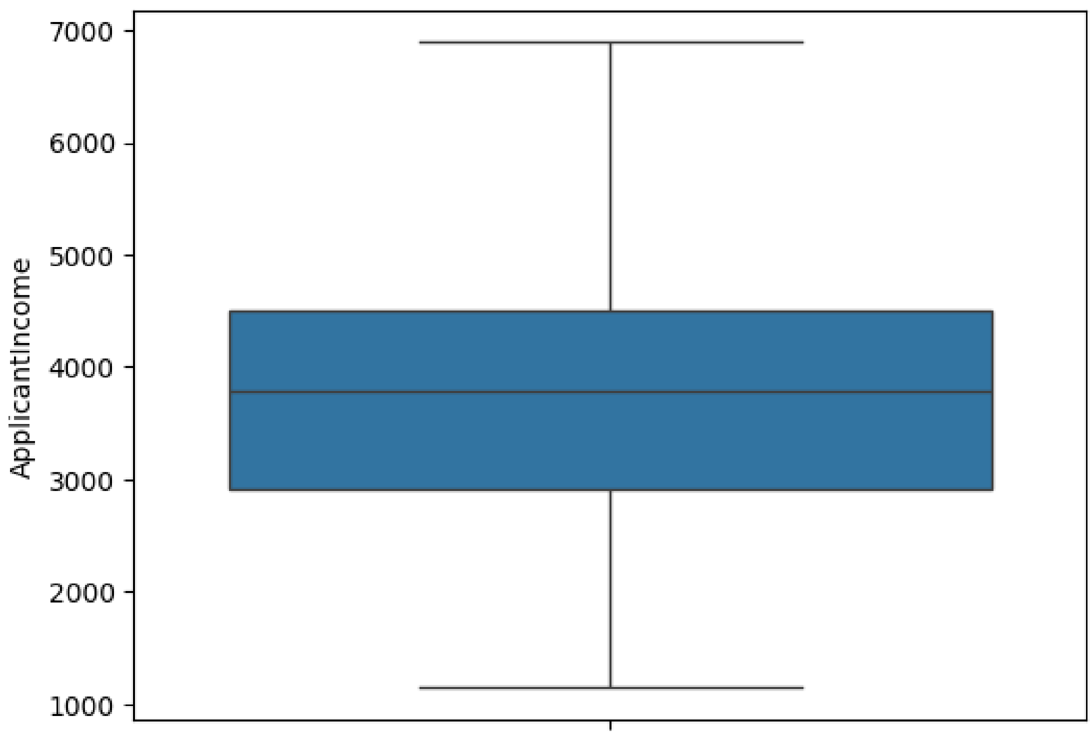
```
[ ]: # Defining new outlier boundaries.

     lower_bound = Q1 - 1.5 * IQR
```

```
print(lower_bound)

upper_bound = Q3 + 1.5 * IQR
print(upper_bound)
```

```
529.0
6893.0
```

[ ]:
```
# Now, instead of replacing with Median, Capping the outliers to boundary␣
↪values.

df['ApplicantIncome'] = np.where(df['ApplicantIncome'] < lower_bound,␣
↪lower_bound, df['ApplicantIncome'])
df['ApplicantIncome'] = np.where(df['ApplicantIncome'] > upper_bound,␣
↪upper_bound, df['ApplicantIncome'])
```

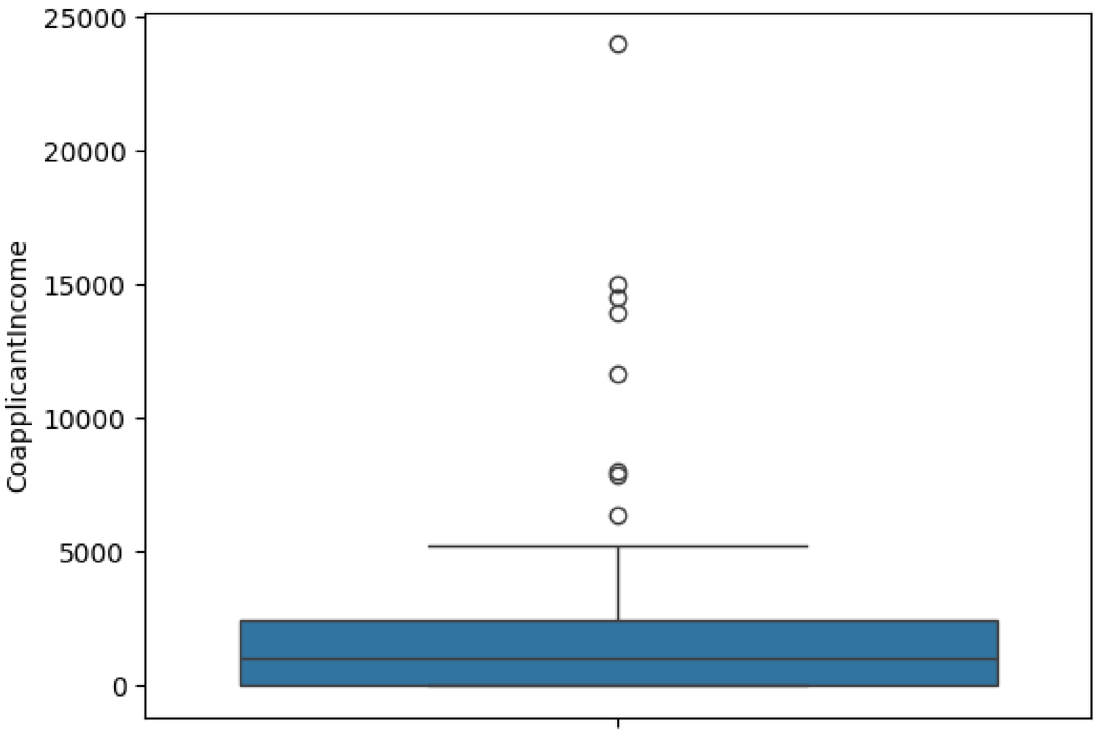
[ ]:
```
# Checking our Distribution again.

sns.boxplot(df['ApplicantIncome'])
plt.show()
```



[ ]:
```
'''Feature - CoapplicantIncome  '''
```

```python
# Drawing a boxplot of feature 'CoapplicantIncome' for checking if there's any
↪outliers exist

sns.boxplot(df['CoapplicantIncome'])
```

[ ]: <Axes: ylabel='CoapplicantIncome'>



[ ]:
```python
'''Outliers handling in feature 'CoapplicantIncome' by IQR Method.'''
```

[ ]:
```python
# Calculating IQR for 'CoapplicantIncome'.

Q1 = df['CoapplicantIncome'].quantile(0.25)
print(f"Q1 is {Q1}")

Q3 = df['CoapplicantIncome'].quantile(0.75)
print(f"Q3 is {Q3}")
```

```
Q1 is 0.0
Q3 is 2430.5
```

[ ]:
```python
IQR = Q3 - Q1
print(f"IQR is {IQR}")
```

```
IQR is 2430.5
```

[ ]:
```
# Defining the outlier boundaries.

lower_bound = Q1 - 1.5 * IQR
print(lower_bound)

upper_bound = Q3 + 1.5 * IQR
print(upper_bound)
```

```
-3645.75
6076.25
```

[ ]:
```
# Identifying outliers.

# Our dataset lies between -3645.75 and 6076.25 as per IQR Method, therefore␣
↪any value below -3645.75 and beyond 6076.25 is considered as an outlier in␣
↪feature 'CoapplicantIncome'.

outliers = df[(df['CoapplicantIncome'] < lower_bound) |␣
↪(df['CoapplicantIncome'] > upper_bound)]
outliers
```

[ ]:
```
      Loan_ID  Gender Married  Dependents     Education Self_Employed  \
18   LP001108    Male     Yes         0.0      Graduate            No
25   LP001153    Male      No         0.0      Graduate            No
85   LP001452    Male     Yes         2.0      Graduate            No
123  LP001652    Male      No         0.0      Graduate            No
230  LP002294    Male      No         0.0      Graduate            No
237  LP002325    Male     Yes         2.0  Not Graduate            No
284  LP002570  Female     Yes         2.0      Graduate            No
351  LP002901    Male      No         0.0      Graduate            No

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
18            3786.0               7916       125.0             360.0
25            3786.0              24000       148.0             360.0
85            4635.0               8000       102.0             360.0
123           2500.0               6414       187.0             360.0
230           3786.0              14507       130.0             360.0
237           6166.0              13983       102.0             360.0
284           3786.0              11666       125.0             360.0
351           2283.0              15000       106.0             360.0

     Credit_History Property_Area
18                1         Urban
25                0         Rural
85                1         Rural
```

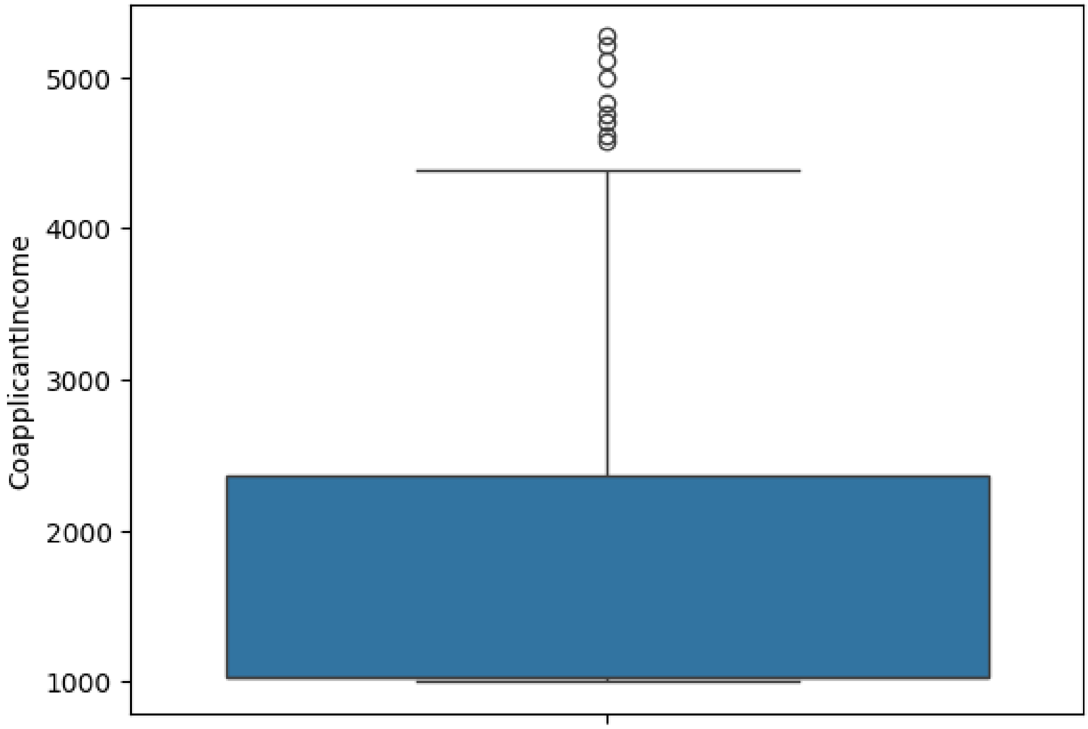
```
123                0        Rural
230                1    Semiurban
237                1        Rural
284                1        Urban
351                1        Rural
```

[ ]:
```
# Checking Median.

median_CoapplicantIncome = df['CoapplicantIncome'].median()
print(median_CoapplicantIncome)
```

```
1025.0
```

[ ]:
```
# Replacing Outliers in 'CoapplicantIncome' with Median
# and
# Replacing 'CoapplicantIncome' < 1000 with Median too.
# (Reason being many co-applicant have either 0 or barely have any income,␣
↪which is practically not a possible income for a co-applicant)

df['CoapplicantIncome'] = np.where((df['CoapplicantIncome'] < lower_bound) |␣
↪(df['CoapplicantIncome'] > upper_bound), median_CoapplicantIncome,␣
↪df['CoapplicantIncome'])

df['CoapplicantIncome'] = np.where(df['CoapplicantIncome'] < 1000,␣
↪median_CoapplicantIncome, df['CoapplicantIncome'])
```

[ ]:
```
# We see that there are still a few outliers exists.

sns.boxplot(df['CoapplicantIncome'])
plt.show()
```




```
[ ]: # Calculate the IQR for 'CoapplicantIncome' again after the previous␣
     ↪transformations.

     Q1 = df['CoapplicantIncome'].quantile(0.25)
     print(f"Q1 is {Q1}")

     Q3 = df['CoapplicantIncome'].quantile(0.75)
     print(f"Q3 is {Q3}")
```

```
Q1 is 1025.0
Q3 is 2368.5
```

```
[ ]: IQR = Q3 - Q1
     print(f"IQR is {IQR}")
```

```
IQR is 1343.5
```

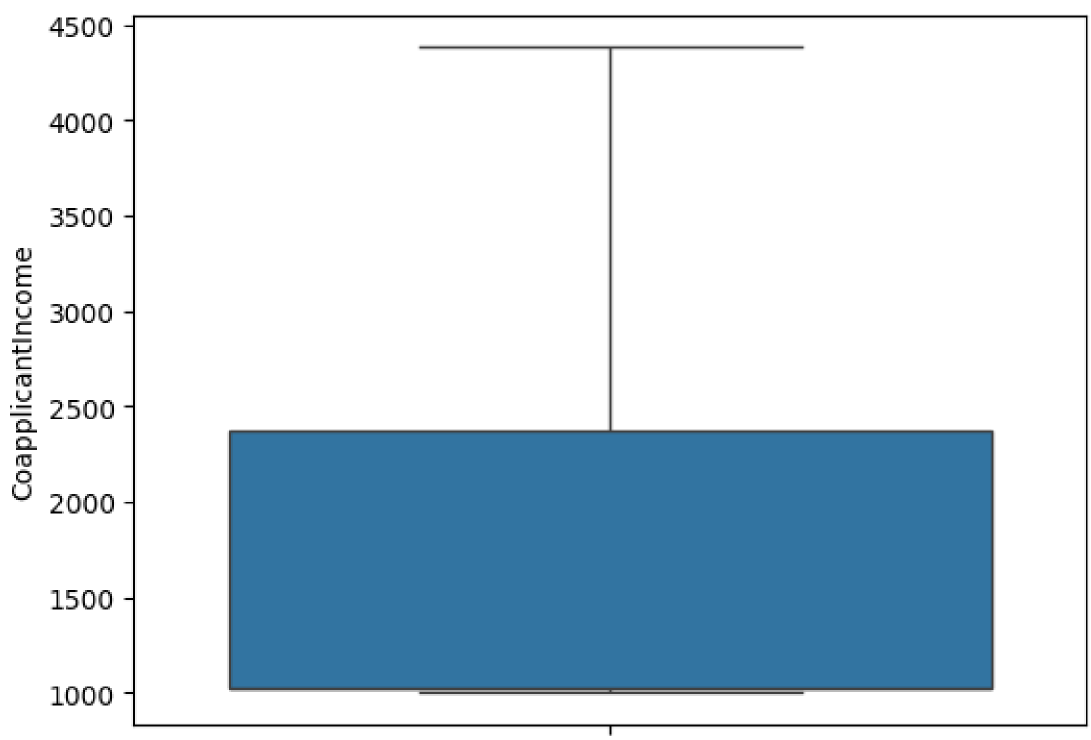
```
[ ]: # Defining new outlier boundaries.

     lower_bound = Q1 - 1.5 * IQR
     print(lower_bound)

     upper_bound = Q3 + 1.5 * IQR
```

```
print(upper_bound)
```

```
-990.25
4383.75
```

[ ]:
```
# Now, instead of replacing with Median, Capping the outliers to boundary␣
↪values.

df['CoapplicantIncome'] = np.where(df['CoapplicantIncome'] < lower_bound,␣
↪lower_bound, df['CoapplicantIncome'])
df['CoapplicantIncome'] = np.where(df['CoapplicantIncome'] > upper_bound,␣
↪upper_bound, df['CoapplicantIncome'])
```

[ ]:
```
# Visualizing the Distribution after handling outliers

sns.boxplot(df['CoapplicantIncome'])
plt.show()
```



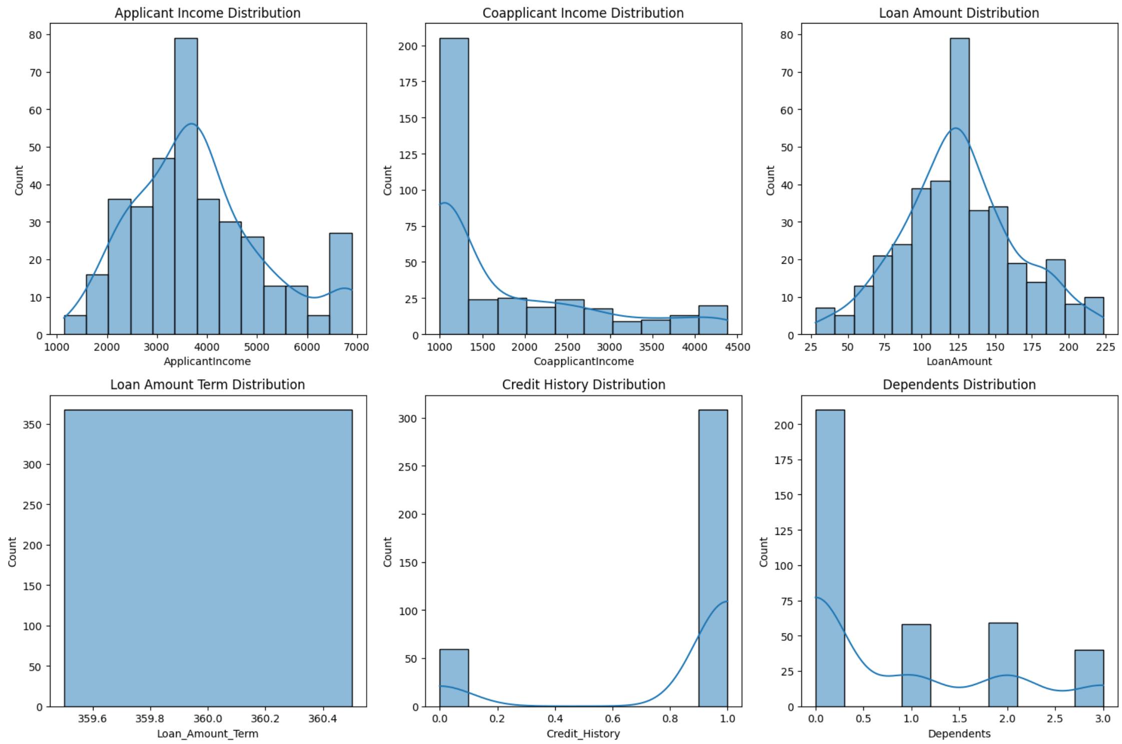
[ ]:
```
df.describe()
```

[ ]:
```
       Dependents  ApplicantIncome  CoapplicantIncome  LoanAmount  \
count  367.000000       367.000000         367.000000  367.000000
mean     0.806540      3839.727520        1766.116485  126.074932
```

```
std       1.065177      1332.092529        1028.615359   39.314453
min       0.000000      1141.000000        1000.000000   28.000000
25%       0.000000      2915.500000        1025.000000  101.000000
50%       0.000000      3786.000000        1025.000000  125.000000
75%       2.000000      4506.500000        2368.500000  150.000000
max       3.000000      6893.000000        4383.750000  223.500000

       Loan_Amount_Term  Credit_History
count             367.0      367.000000
mean              360.0        0.839237
std                 0.0        0.367814
min               360.0        0.000000
25%               360.0        1.000000
50%               360.0        1.000000
75%               360.0        1.000000
max               360.0        1.000000
```

[ ]:
```
'''Question 1 - Explore the distribution of numeric columns using the following␣
↪visualizations-'''

'''HISTOGRAMS: Plot the frequency distribution of key Numeric variables.'''

# Plotting Histograms for Numerical Features

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
sns.histplot(df['ApplicantIncome'], kde=True)
plt.title('Applicant Income Distribution')

plt.subplot(2, 3, 2)
sns.histplot(df['CoapplicantIncome'], kde=True)
plt.title('Coapplicant Income Distribution')

plt.subplot(2, 3, 3)
sns.histplot(df['LoanAmount'], kde=True)
plt.title('Loan Amount Distribution')

plt.subplot(2, 3, 4)
sns.histplot(df['Loan_Amount_Term'], kde=True)
plt.title('Loan Amount Term Distribution')

plt.subplot(2, 3, 5)
sns.histplot(df['Credit_History'], kde=True)
plt.title('Credit History Distribution')

plt.subplot(2, 3, 6)
```

```python
sns.histplot(df['Dependents'], kde=True)
plt.title('Dependents Distribution')

plt.tight_layout()
plt.show()
```

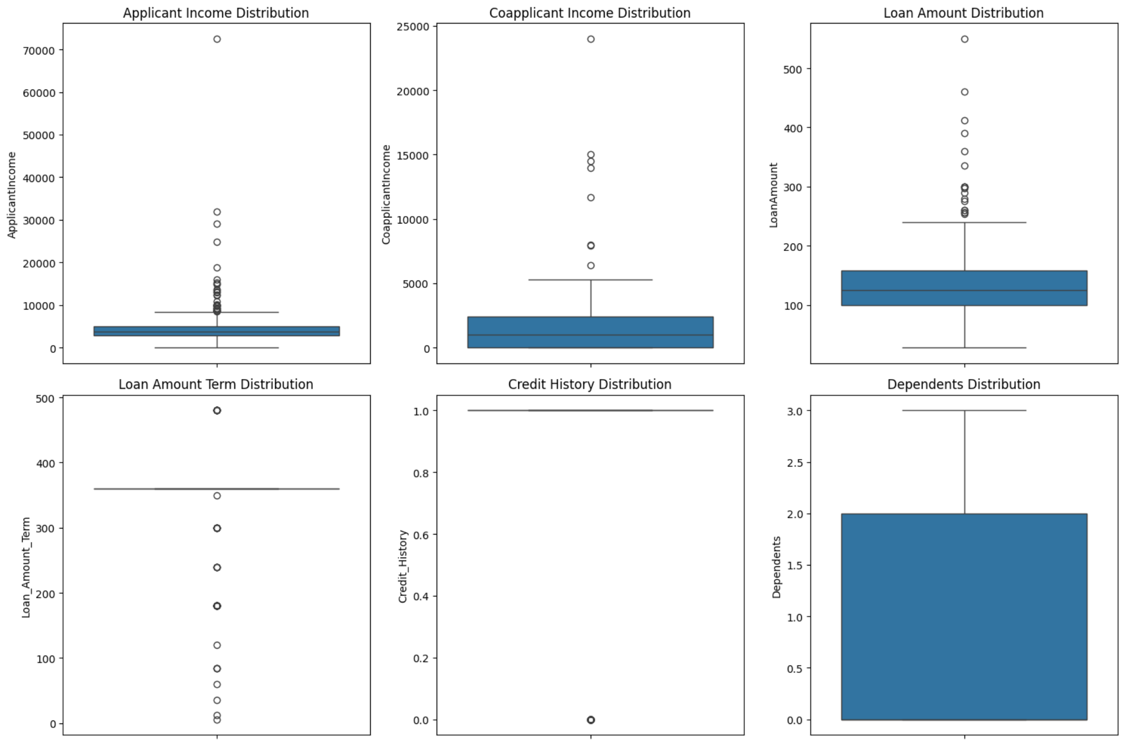



```python
[ ]: '''Meaningful insights from Question 1 (Histogram)'''

# 1. Applicant Income: The distribution is right-skewed, indicating a higher concentration of applicants with lower incomes.
# This suggests that the majority of loan applicants come from a specific income bracket.

# 2. Coapplicant Income: Similar to applicant income, the coapplicant income distribution is also right-skewed,
# Implying that most co-applicants also have lower incomes.

# 3. Loan Amount: The loan amount distribution shows a peak around a certain value, indicating a common loan amount requested by applicants.
# This could reflect the average affordability or typical loan requirements in the market.
```

```
# 4. Loan Amount Term: The loan amount term distribution reveals the most␣
↪frequent loan durations chosen by applicants.
# This information can help understand the preferred repayment timelines for␣
↪loans.

# 5. Credit History: The credit history distribution shows a clear distinction␣
↪between applicants with and without a credit history.
# This highlights the importance of credit history in loan approval decisions.

# 6. Dependents: The dependents distribution indicates the number of dependents␣
↪most commonly declared by applicants.
# This information can provide insights into the family structures of loan␣
↪applicants.

# These insights provide a comprehensive understanding of the distribution of␣
↪key numeric variables, enabling better decision-making in loan approval␣
↪processes.
```

[ ]:
```
'''Question 1 - Explore the distribution of numeric columns using the following␣
↪visualizations-'''

'''Box Plots: Identify potential outliers and visualize the spread of data.'''

# Plotting Box Plots for Numerical Features

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
sns.boxplot(df['ApplicantIncome'])
plt.title('Applicant Income Distribution')

plt.subplot(2, 3, 2)
sns.boxplot(df['CoapplicantIncome'])
plt.title('Coapplicant Income Distribution')

plt.subplot(2, 3, 3)
sns.boxplot(df['LoanAmount'])
plt.title('Loan Amount Distribution')

plt.subplot(2, 3, 4)
sns.boxplot(df['Loan_Amount_Term'])
plt.title('Loan Amount Term Distribution')

plt.subplot(2, 3, 5)
sns.boxplot(df['Credit_History'])
plt.title('Credit History Distribution')
```

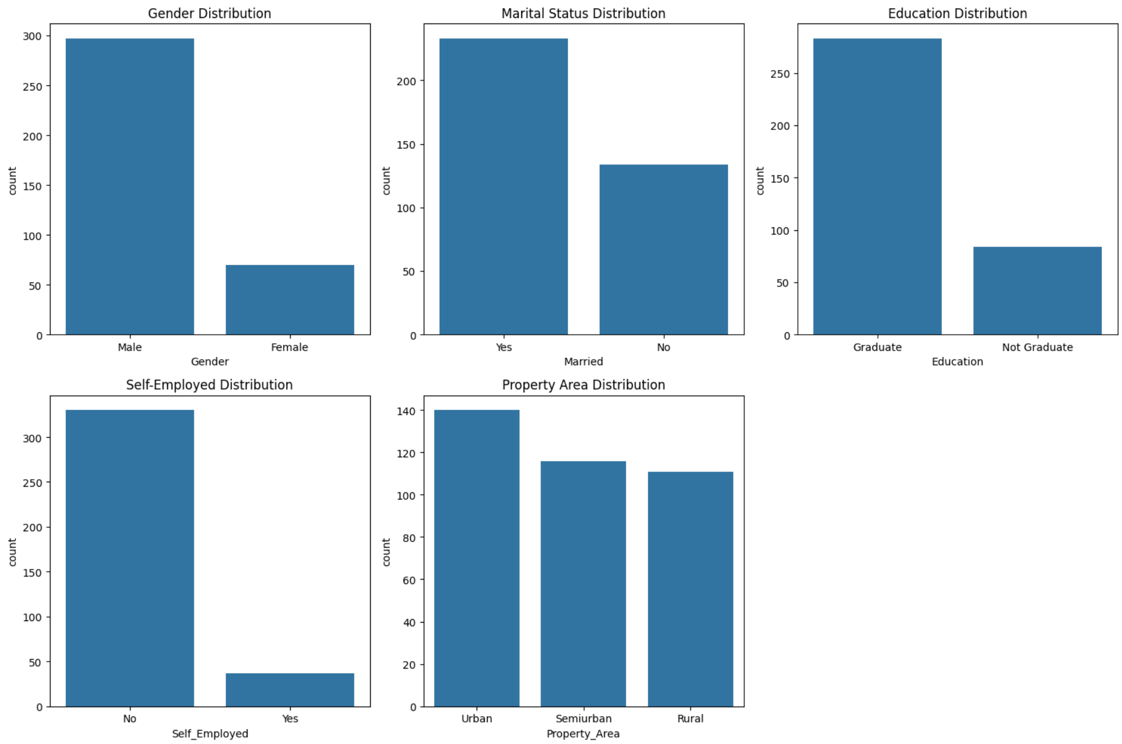
```
plt.subplot(2, 3, 6)
sns.boxplot(df['Dependents'])
plt.title('Dependents Distribution')

plt.tight_layout()
plt.show()
```



[ ]:
```
'''Meaningful insights from Question 1 (Box Plots)'''

# 1. Applicant Income: The box plot shows a number of outliers on the higher␣
↪end of the income spectrum, suggesting there are some applicants with␣
↪significantly higher incomes than the majority. The median income appears to␣
↪be around  3,800.

# 2. Coapplicant Income: Similar to ApplicantIncome, there are outliers␣
↪indicating some co-applicants have substantially higher incomes.
# The median co-applicant income seems to be lower than the applicant's, around␣
↪ 1,500.

# 3. Loan Amount: The distribution of loan amounts is relatively symmetrical,␣
↪with a median value around  128.
```

```
# There are a few outliers on the higher end, indicating some individuals are␣
↪applying for larger loans.

# 4. Loan Amount Term: Most loan terms are clustered around 360 months (30␣
↪years), with a few outliers representing shorter-term loans.

# 5. Credit History: This plot clearly shows the majority of applicants have a␣
↪credit history (value of 1). There's a smaller group with no credit history␣
↪(value of 0).
```

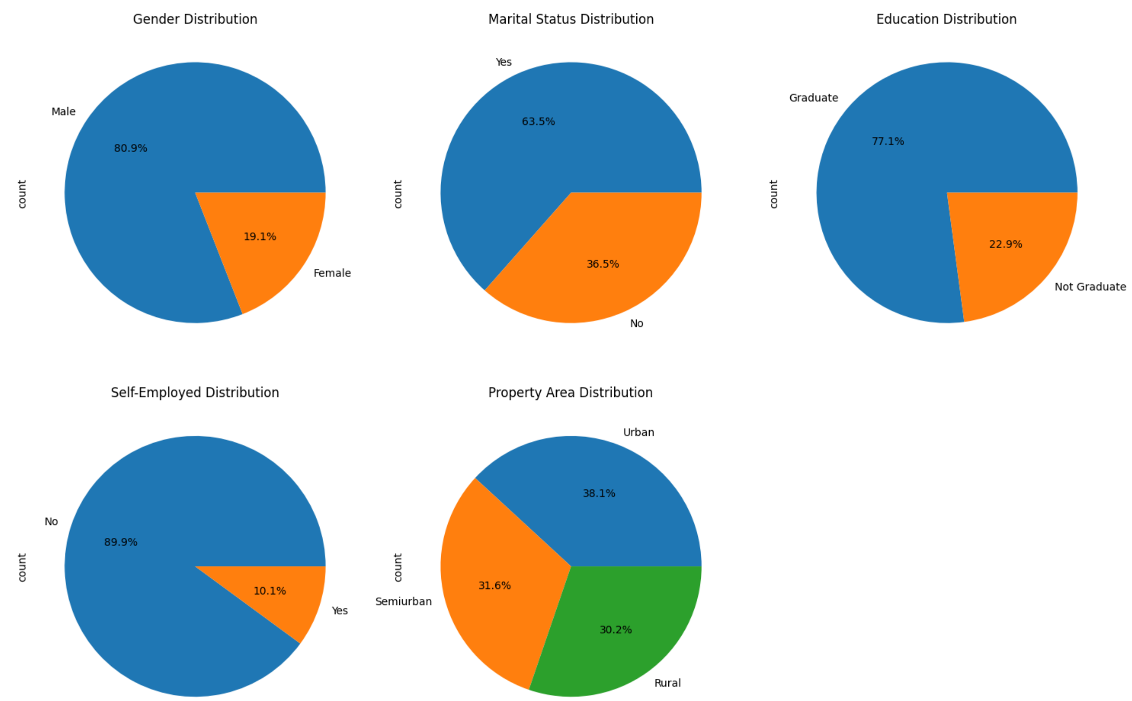
[ ]:
```
'''Question 2 - Analyze categorical variables by creating the following␣
↪plots-'''

'''Bar Charts: Visualize the frequency distribution of categorical variables.'''

# Plotting Bar Charts for Categorical Features

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
sns.countplot(x='Gender', data=df)
plt.title('Gender Distribution')

plt.subplot(2, 3, 2)
sns.countplot(x='Married', data=df)
plt.title('Marital Status Distribution')

plt.subplot(2, 3, 3)
sns.countplot(x='Education', data=df)
plt.title('Education Distribution')

plt.subplot(2, 3, 4)
sns.countplot(x='Self_Employed', data=df)
plt.title('Self-Employed Distribution')

plt.subplot(2, 3, 5)
sns.countplot(x='Property_Area', data=df)
plt.title('Property Area Distribution')

plt.tight_layout()
plt.show()
```



[ ]:
```
'''Meaningful insights from Question 2 (Bar Charts)'''

# 1. Gender: The majority of loan applicants are male, indicating a potential gender bias in loan applications.
# This could be further investigated to understand the reasons behind this disparity.

# 2. Marital Status: A significant portion of loan applicants are married, suggesting that married individuals may have a higher demand for loans.
# This could be attributed to factors like family responsibilities or joint financial planning.

# 3. Education: Most loan applicants are graduates, implying that higher education levels might be associated with increased loan applications.
# This could reflect a greater awareness of financial products or a higher need for loans among educated individuals.

# 4. Self-Employed: The majority of loan applicants are not self-employed, indicating that salaried individuals constitute a larger portion of loan seekers.
# This could be due to the perceived stability of salaried income compared to self-employment.
```

```
# 5. Property Area: The distribution of property areas shows a relatively even␣
↪distribution across urban, semi-urban, and rural areas.
# This suggests that loan applications are not significantly concentrated in␣
↪any specific type of property area.

# These insights provide valuable information about the characteristics of loan␣
↪applicants based on categorical variables, enabling a more targeted approach␣
↪to loan approval and risk assessment.
```

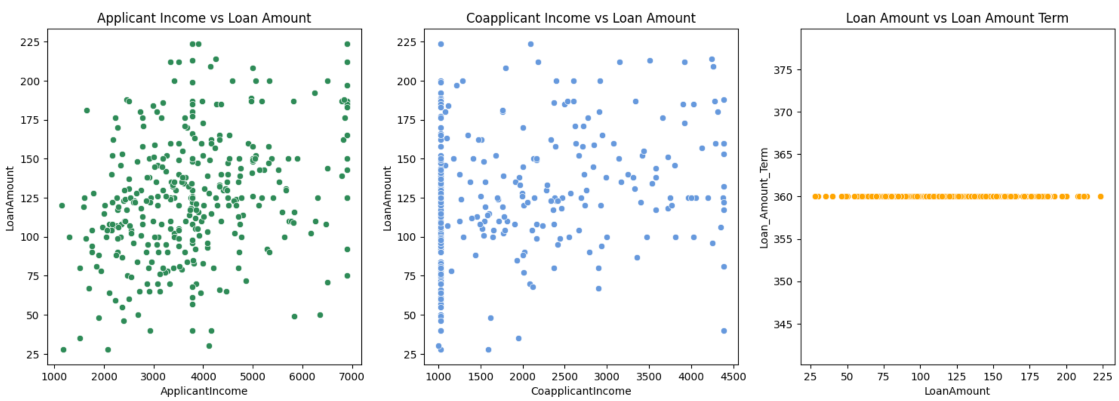
[ ]:
```
'''Question 2 - Analyze categorical variables by creating the following␣
↪plots-'''

'''Pie Charts: Represent the composition of categorical variables.'''

# Plotting Pie Charts for Categorical Features

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
df['Gender'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Gender Distribution')

plt.subplot(2, 3, 2)
df['Married'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Marital Status Distribution')

plt.subplot(2, 3, 3)
df['Education'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Education Distribution')

plt.subplot(2, 3, 4)
df['Self_Employed'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Self-Employed Distribution')

plt.subplot(2, 3, 5)
df['Property_Area'].value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Property Area Distribution')

plt.tight_layout()
plt.show()
```



[ ]:
```
'''Meaningful insights from Question 2 (Pie Charts)'''

# Gender Distribution: Around 81% of the loan applicants are male, indicating a significant gender imbalance in loan applications.

# Marital Status Distribution: Approximately 65% of the applicants are married, suggesting that married individuals may be more likely to apply for loans.

# Education Distribution: A majority (around 78%) of the applicants are graduates, indicating a higher likelihood of loan applications from individuals with higher education levels.

# Self-Employed Distribution: Only about 14% of the applicants are self-employed, implying that a majority of loan applications come from salaried individuals.

# Property Area Distribution: The distribution of applicants across property areas (Semiurban, Urban, Rural) is relatively balanced, with semiurban areas having a slightly higher proportion (around 38%).
```

[ ]:
```
'''Question 3 - Create scatter plots to explore relationships between pairs of numeric variables.'''
```

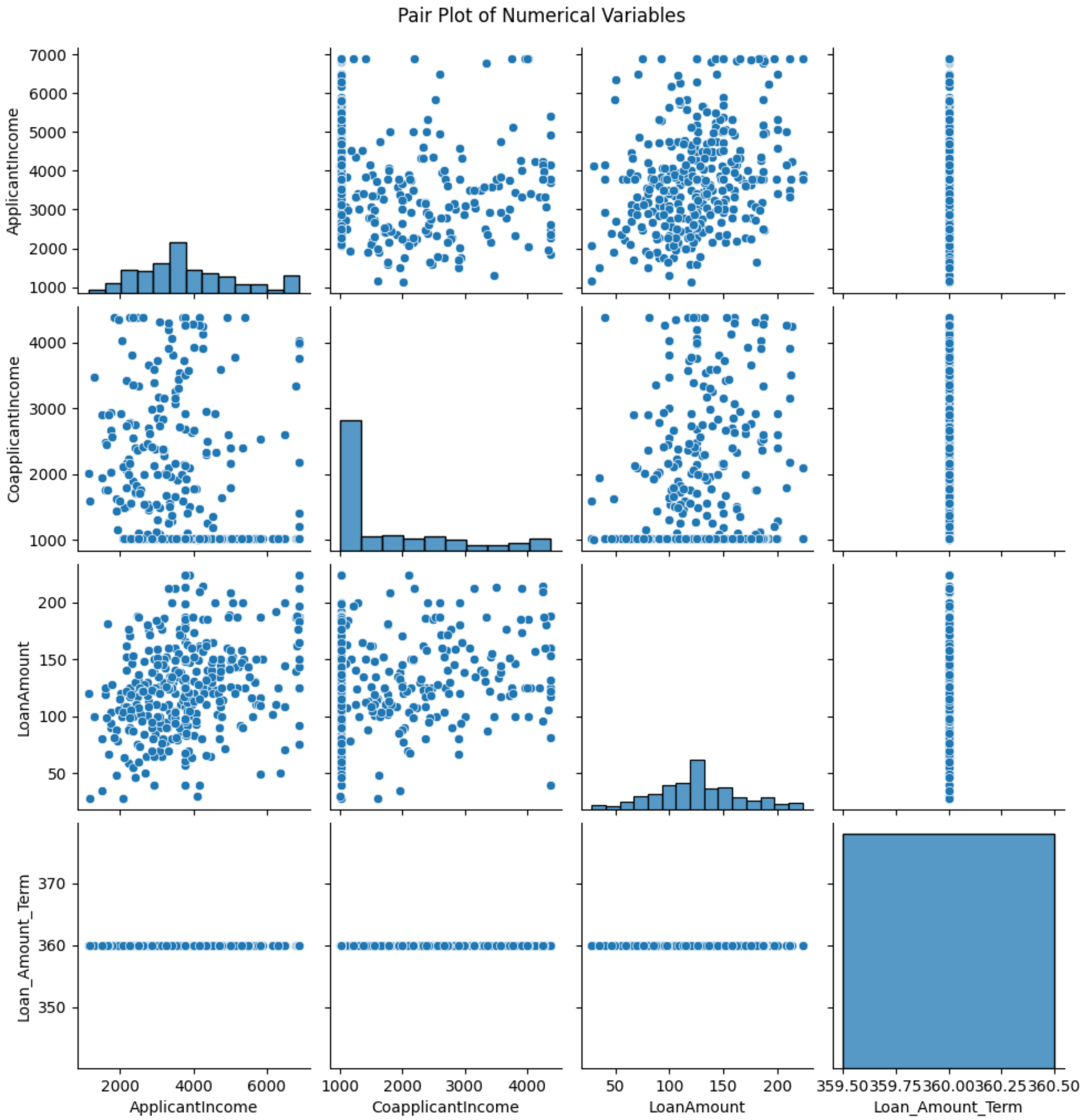
```
# Plotting Scatter Plots for Numerical variables

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
sns.scatterplot(x='ApplicantIncome', y='LoanAmount', color = 'seagreen',␣
↪data=df)
plt.title('Applicant Income vs Loan Amount')

plt.subplot(2, 3, 2)
sns.scatterplot(x='CoapplicantIncome', y='LoanAmount', color = '#69d', data=df)
plt.title('Coapplicant Income vs Loan Amount')

plt.subplot(2, 3, 3)
sns.scatterplot(x='LoanAmount', y='Loan_Amount_Term', color = 'orange', data=df)
plt.title('Loan Amount vs Loan Amount Term')

plt.tight_layout()
plt.show()
```



[ ]:
```
'''Meaningful insights from Question 3'''

# 1. Applicant Income vs Loan Amount:
# There's a slight positive correlation, indicating higher-income applicants␣
↪tend to request larger loans.
# However, the correlation isn't very strong, suggesting other factors␣
↪influence loan amount.

# 2. Coapplicant Income vs Loan Amount:
# A weaker positive correlation than with Applicant Income, suggesting␣
↪coapplicant income plays a less significant role in loan amount␣
↪determination.
```

```
# 3. Loan Amount vs Loan Amount Term:
# No clear correlation is observed, indicating loan amount and term are largely␣
↪independent of each other.
# This suggests that loan term is determined based on factors other than the␣
↪loan amount.
```

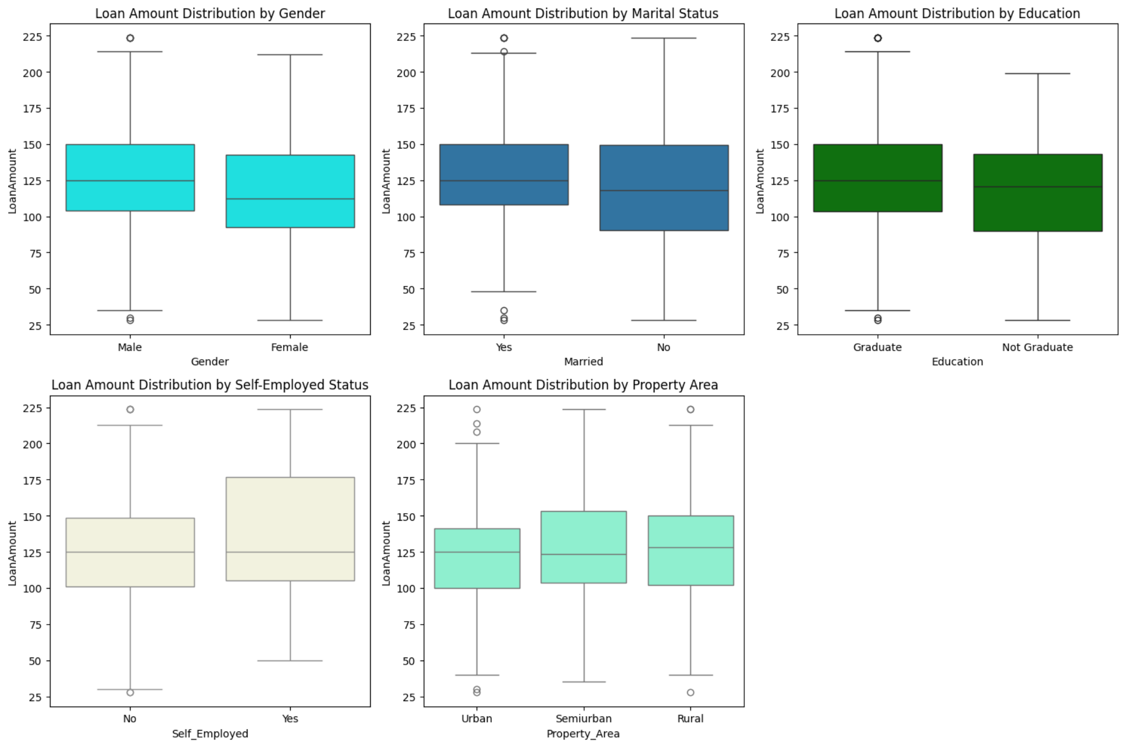
[ ]:
```
'''Question 4 - Use pair plots (scatter matrix) to visualize interactions␣
↪between multiple numeric variables simultaneously.'''

# Plotting Pair Plots for Numerical variables

sns.pairplot(df[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',␣
↪'Loan_Amount_Term']])
plt.suptitle('Pair Plot of Numerical Variables', y=1.02)
plt.show()
```



[ ]:
```
'''Meaningful insights from Question 4'''

# 1. There's a positive correlation between 'ApplicantIncome' and 'LoanAmount'.
 ↪As 'ApplicantIncome' increases, the 'LoanAmount' they are eligible for also
 ↪tends to increase.

# 2. There's no significant correlation between 'CoapplicantIncome' and
 ↪'LoanAmount'. The 'LoanAmount' doesn't seem to be strongly influenced by the
 ↪'CoapplicantIncome'.
```

```
# 3. There's no clear correlation between 'LoanAmount' and 'Loan_Amount_Term'.␣
↪The duration of the loan doesn't appear to have a strong relationship with␣
↪the amount borrowed.
```

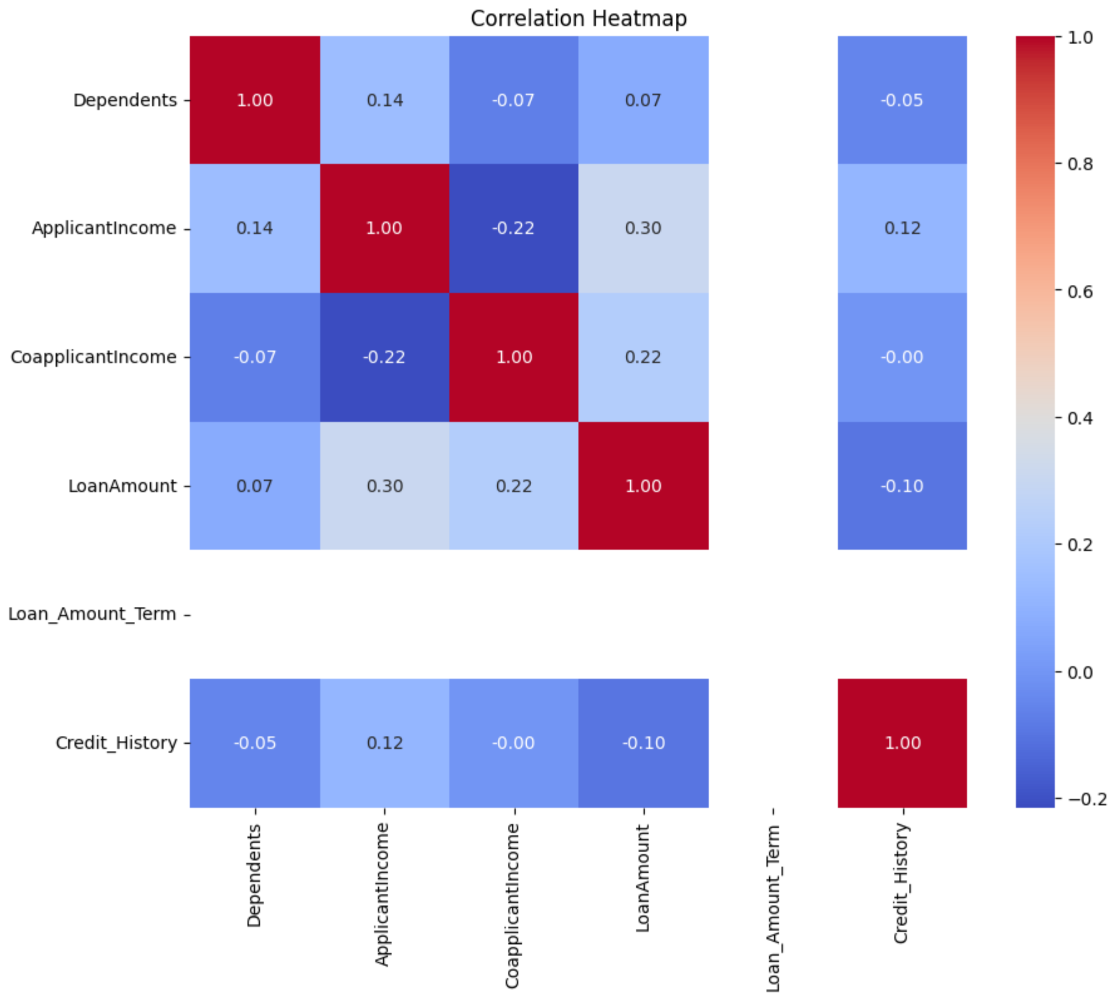
[ ]:
```
'''Question 5 - Investigate the relationship between categorical and numeric␣
↪variables using Box plots.'''

# Plotting Box Plots for Categorical vs Numerical variables

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
sns.boxplot(x='Gender', y='LoanAmount', color='Cyan', data=df)
plt.title('Loan Amount Distribution by Gender')

plt.subplot(2, 3, 2)
sns.boxplot(x='Married', y='LoanAmount', data=df)
plt.title('Loan Amount Distribution by Marital Status')

plt.subplot(2, 3, 3)
sns.boxplot(x='Education', y='LoanAmount', color='Green', data=df)
plt.title('Loan Amount Distribution by Education')

plt.subplot(2, 3, 4)
sns.boxplot(x='Self_Employed', y='LoanAmount', color = 'beige', data=df)
plt.title('Loan Amount Distribution by Self-Employed Status')

plt.subplot(2, 3, 5)
sns.boxplot(x='Property_Area', y='LoanAmount', color = 'aquamarine', data=df)
plt.title('Loan Amount Distribution by Property Area')

plt.tight_layout()
plt.show()
```



[ ]:
```
'''Meaningful insights from Question 5'''

# 1. There's no significant difference in 'LoanAmount' based on 'Gender'. Both
↪males and females tend to apply for similar loan amounts.

# 2. Married individuals tend to apply for slightly higher 'LoanAmount'
↪compared to unmarried individuals.

# 3. Graduates tend to apply for higher 'LoanAmount' compared to non-graduates.
↪This suggests that higher education might be associated with greater
↪financial needs or borrowing capacity.

# 4. There's no substantial difference in 'LoanAmount' between self-employed
↪and non-self-employed individuals.

# 5. 'LoanAmount' distribution varies slightly across different 'Property_Area'.
↪ Applicants from semiurban areas tend to apply for slightly higher loan
↪amounts compared to those from urban or rural areas.
```

[ ]:
```
''' Question 6 - Perform a correlation analysis to identify relationships
↪between numeric variables. Visualize correlations using a heatmap.'''

# Selecting only numeric columns for correlation analysis
```

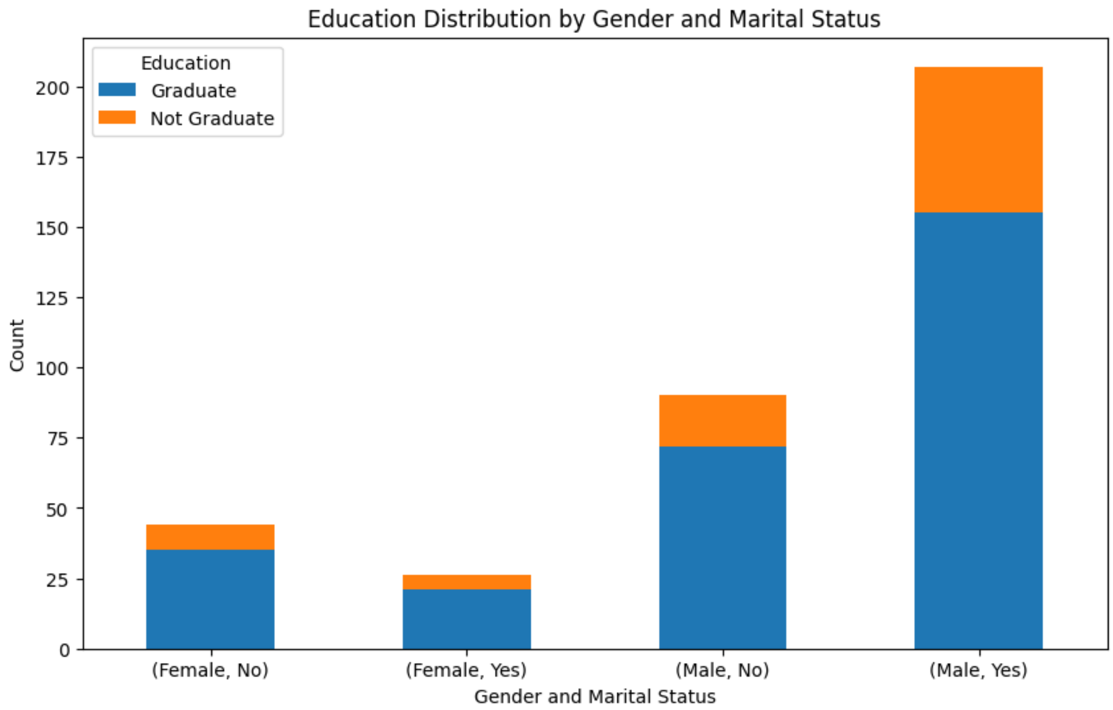
```python
numeric_df = df.select_dtypes(include=['number'])

# Calculating the correlation matrix for numerical columns

correlation_matrix = numeric_df.corr()

# Ploting heatmap

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



[ ]:
```python
'''Meaningful insights from Question 6'''
```

```
# 1. Positive Correlation between Loan Amount and Applicant Income:
#    - The heatmap shows a moderate positive correlation (0.57) between␣
↪'LoanAmount' and 'ApplicantIncome'.
#    - This suggests that individuals with higher incomes tend to apply for␣
↪larger loans.

# 2. Weak Correlation between Loan Amount and Coapplicant Income:
#    - The correlation between 'LoanAmount' and 'CoapplicantIncome' is␣
↪relatively weak (0.19).
#    - This indicates that the coapplicant's income has a lesser impact on the␣
↪loan amount compared to the applicant's income.

# 3. No Strong Linear Relationships:
#    - There are no extremely strong linear relationships (close to 1 or -1)␣
↪observed in the heatmap.
#    - This implies that the relationships between these numerical variables␣
↪are not strictly linear and might involve other factors.

# 4. Potential Multicollinearity:
#    - While not extremely high, the correlation between 'ApplicantIncome' and␣
↪'CoapplicantIncome' (0.38) suggests a degree of multicollinearity.
#    - This could be considered during feature selection for modeling,␣
↪especially if using linear models sensitive to multicollinearity.
```

[ ]:
```
'''Question 7 - Create a stacked bar chart to show the distribution of␣
↪categorical variables across multiple categories.'''

# Assuming you want to visualize the distribution of 'Education' across␣
↪'Gender' and 'Married',

# Grouping the data and calculating counts

grouped_data = df.groupby(['Gender', 'Married', 'Education'])['Education'].
↪count().unstack()

# Ploting the stacked bar chart

grouped_data.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.title('Education Distribution by Gender and Marital Status')
plt.xlabel('Gender and Marital Status')
plt.ylabel('Count')
plt.legend(title='Education')
plt.xticks(rotation=0)
plt.show()
```



[ ]:
```
'''Meaningful insights from Question 7'''

# 1. Education and Gender:
#    - A higher proportion of males across both married and unmarried categories have a graduate degree compared to females.
#    - The difference in graduate education between genders is more pronounced in the married category.

# 2. Education and Marital Status:
#    - In both male and female categories, a higher proportion of married individuals have a graduate degree compared to unmarried individuals.
#    - This suggests a possible correlation between higher education and the likelihood of getting married.

# 3. Overall Trend:
#    - The majority of loan applicants, regardless of gender or marital status, have a graduate degree.
#    - This indicates that higher education might be a common factor among those seeking loans.

# These insights can be further investigated to understand the impact of education on loan approval rates and other relevant factors.
```

[ ]:
```
'''FINAL WORDINGS - Summarize the key findings, draw conclusions, and provide␣
↪recommendations based on the insights gained from the analysis'''

'''
Key Findings:

1. Distribution of Numeric Variables:

  - ApplicantIncome and CoapplicantIncome are right-skewed, indicating a␣
↪concentration of lower incomes with a few high earners.
  - LoanAmount shows a relatively normal distribution with some outliers.
  - Loan_Amount_Term is predominantly concentrated at 360 months.
  - Credit_History is negatively skewed, suggesting most applicants have a␣
↪credit history.
  - Dependents distribution indicates a higher proportion of applicants with no␣
↪dependents.

2. Categorical Variable Analysis:

  - Majority of applicants are male and married.
  - Most applicants are graduates and not self-employed.
  - Property_Area distribution is relatively balanced across urban, semiurban,␣
↪and rural areas.

3. Relationships between Variables:

  - Positive correlation between ApplicantIncome and LoanAmount, suggesting␣
↪higher income applicants tend to request larger loans.
  - Weak positive correlation between CoapplicantIncome and LoanAmount.
  - No strong linear relationship between LoanAmount and Loan_Amount_Term.
  - Box plots reveal variations in LoanAmount distribution across different␣
↪categories of Gender, Marital Status, Education, Self_Employed, and␣
↪Property_Area.

Conclusions:

- Income and Loan Amount: Applicant income plays a significant role in␣
↪determining the loan amount.
- Demographic Factors: Gender, marital status, education, and employment status␣
↪influence loan application characteristics.
- Credit History: A good credit history is prevalent among applicants.

Recommendations:

  - Target Marketing: Tailor loan products and marketing strategies based on␣
↪income levels and demographic characteristics.
```

```
 - Risk Assessment: Consider income, credit history, and other factors for␣
↪loan approval and risk assessment.
 - Product Diversification: Offer loan products with varying terms and amounts␣
↪to cater to different customer needs.
 - Further Analysis: Explore additional factors and interactions to gain␣
↪deeper insights and refine decision-making.

'''
```

[ ]:
```
#'''THANK YOU FOR YOUR VALUABLE TIME'''#
```